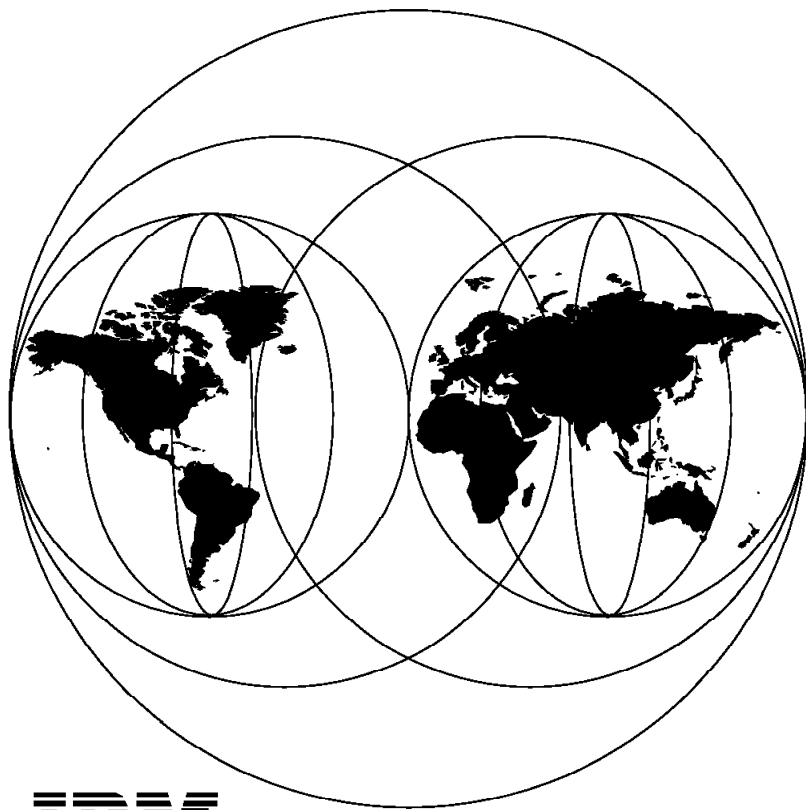


Integrating VisualAge Generator and FlowMark

June 1997



**International Technical Support Organization
San Jose Center**



International Technical Support Organization

SG24-4239-00

Integrating VisualAge Generator and FlowMark

June 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, "Special Notices" on page 47.

First Edition (June 1997)

This edition applies to:

- Version 2.3 of IBM FlowMark, Program Number 5697-216, for use with the OS/2 Warp V3 and Windows NT V3.51 operating systems
- Version 2.2 of IBM VisualAge Generator Developer for OS/2, Program Number 5622-580, for use with the OS/2 operating system
- Version 2.2 of IBM VisualAge Generator Workgroup Services for OS/2, AIX, and Windows, Program Number 5622-585, for use with the OS/2, AIX, and Windows operating systems

Comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1997. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	v
The Team That Wrote This Redbook	v
Comments Welcome	vi
 Chapter 1. VisualAge Generator and FlowMark	 1
1.1 VisualAge Generator	2
1.2 FlowMark	3
1.3 FlowMark Workflow Manager	6
1.4 VisualAge Generator Integration Issues	7
 Chapter 2. Integrating VisualAge Generator with FlowMark	 11
2.1 FlowMark Integration	11
2.1.1 Objectives	11
2.1.2 Options	12
2.2 Communication Using a Dynamic Link Library	13
2.2.1 STARTGUI.EXE	14
2.2.2 VG4FMDLL.DLL	14
2.2.3 GUI Application	14
2.2.4 File-Based Communication	15
2.3 Communication Using Dynamic Data Exchange	15
2.3.1 STARTDDE.EXE	16
2.3.2 GUI Application	17
2.3.3 DDE-Based Communication	17
 Chapter 3. Implementing a FlowMark-to-VisualAge Generator Interface	 19
3.1 FlowMark Workflow Manager Definitions	19
3.2 FlowMark-VisualAge Generator Initialization Program	22
3.3 FlowMark-VisualAge Generator Interface EXE Program	22
3.4 FlowMark-Enabled GUI Application	26
3.5 FlowMark-VisualAge Generator Interface DLL	27
3.6 FlowMark-VisualAge Generator Interface Communication Record	29
 Chapter 4. VisualAge Generator and FlowMark Sample Application	 33
4.1 Prerequisite Software	33
4.2 Implementation	34
4.3 Runtime Operation	42
 Appendix A. GUI Application Terms and Implications	 45
 Appendix B. Special Notices	 47
 Appendix C. Related Publications	 49
C.1 International Technical Support Organization Publications	49
C.2 Redbooks on CD-ROMs	49
C.3 Other Publications	49
 How to Get ITSO Redbooks	 51
How IBM Employees Can Get ITSO Redbooks	51

How Customers Can Get ITSO Redbooks	53
IBM Redbook Order Form	54
Index	55
ITSO Redbook Evaluation	57

Preface

This redbook will help you understand the issues of using VisualAge Generator GUIs in a FlowMark workflow process. It will help you develop your own solution. System architects and developers can take advantage of the information about integrating VisualAge Generator with FlowMark to develop solutions to business problems that use both tools. This redbook explains how to design the VisualAge Generator-to-FlowMark interface and includes source and executable code for the sample application used as an example.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization-San Jose Center.

Henrik Vestergaard is a Systems Engineer at IBM Global Services, Professional Services, IBM Denmark. He has been with IBM for eight years, working mostly with the development of client/server applications. He has a master's degree in electrical engineering (computer science) from the Danish Technical University.

Pat McCarthy is a Consulting Application Development Specialist at the International Technical Support Organization-San Jose Center. He writes extensively and teaches IBM classes worldwide on all areas of application development with a specific emphasis on VisualAge Generator and related technologies. Before joining the ITSO in 1990, Pat worked in an IBM internal information systems organization in Poughkeepsie, New York as a programmer, database administrator, and development center leader. Pat received a B.S. in Business Administration from Indiana University of Pennsylvania and an M.S. in Computer Science from Marist College.

Information about the potential of a dynamic data exchange (DDE) integration solution was contributed by:

Dan Ingerslev
IBM Global Services, Professional Services, IBM Denmark.

FlowMark product support and informal redbook reviews were obtained from:

Mike Ebberts
IBM - International Technical Support Organization-Cambridge Center.

Klaus Deinhart
IBM - German Software Development Lab.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 57 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users <http://www.redbooks.ibm.com>

For IBM Intranet users <http://w3.itso.ibm.com>

- Send us a note at the following address:

redbook@vnet.ibm.com

Chapter 1. VisualAge Generator and FlowMark

Workflow management combined with rapid application development and reuse of existing code provide support for effective application system development and deployment in today's business environment. FlowMark and VisualAge Generator are the tools that can be combined (with an appropriate interface) to support those activities and make possible the development and deployment of robust applications that allow a company to achieve a competitive edge.

This redbook discusses the use of VisualAge Generator with FlowMark and shows you why an interface between FlowMark and VisualAge Generator is necessary by explaining the issues of integrating graphical user interface (GUI) applications in a FlowMark process.

We describe how to create an interface between FlowMark and VisualAge Generator and include code samples for you to use in constructing your own solution.

Note: FlowMark may be able to dispatch non-GUI VisualAge Generator applications (text user interface applications or batch applications) as FlowMark managed activities. We did not experiment with those applications.

Our objectives in writing the book were to enable you to understand:

- The way the FlowMark workflow manager works
- The impact on flow control when using VisualAge Generator GUI applications in a FlowMark workflow process
- Problems and advantages of using VisualAge Generator GUI applications in a FlowMark workflow process
- Generic solutions to the problems
- Specific solutions using OS/2 Warp V3, Windows NT V3.51, and Windows NT V4.00
- Code samples that you can include in your own implementation

This redbook is written for people such as system architects and developers who are engaged in developing solutions that include both VisualAge Generator and FlowMark.

To get the most out of this redbook, you should have some knowledge of, and experience with, the development of FlowMark processes and VisualAge Generator GUI applications and the use of the C programming language.

We assume throughout the book that you have this knowledge and experience. However, we include a short description of FlowMark and VisualAge Generator for your review.

Even if you have no knowledge of or experience with development in FlowMark, VisualAge Generator, or C, you may still find useful the discussions on the advantages and disadvantages of using GUI applications in a FlowMark workflow process and the generic solutions we offer. The code samples we provide and reading the *IBM FlowMark: Programming Guide* may even give you an idea of how to implement your own solution.

1.1 VisualAge Generator

VisualAge Generator is a fourth-generation language (4GL) development environment for client/server applications, offering rapid development through its workstation platform development tool (VisualAge Generator Developer). VisualAge Generator Developer provides the capability to define, test, and generate client applications, server applications, and single-system applications. VisualAge Generator supports a variety of operating systems, including OS/2 and Windows NT, and facilitates reuse of existing Cross System Product (CSP) applications.

VisualAge Generator can be used to develop GUI and text user interface (TUI) applications as well as servers and batch applications. These applications can be generated for and used in a variety of runtime environments:

GUI Applications

Can be implemented on OS/2, Windows 3.11, Windows 95, and Windows NT. Generated outputs are used in a VisualAge Generator provided Smalltalk-based runtime environment.

TUIs, Servers, and Batch Applications

Can be implemented on OS/2, Windows NT, AIX, TSO, CICS (MVS, VSE, OS/2, NT, AIX), IMS/DC, MVS/TSO, and AS/400 platforms.

COBOL programs are generated for MVS (CICS, TSO, IMS/DC), VSE CICS, AS/400, and CICS OS/2 runtime environments.

C++ programs are generated for CICS NT, OS/2, Windows NT, AIX, and CICS/6000 platforms.

Generated program execution is supported by the VisualAge Generator Workgroup Services runtime environment.

Several types of GUI applications can be built with VisualAge Generator. In *Object-Based GUI Application Development with VisualGen*, we discuss the types of GUI applications and their role in development of an application system. An extract of that discussion is provided in Appendix A, "GUI Application Terms and Implications" on page 45. The terms described are used in this redbook and are summarized below:

- External GUI applications can be started from a command line and reused.
- Embedded GUI applications can be reused but not started from a command line.
- Functional GUI applications must be external GUI applications and may also reuse other functional GUI applications, external GUI applications, and embedded GUI applications.
- Primary GUI applications must be external GUI applications and may also reuse other functional GUI applications, external GUI applications, and embedded GUI applications.

Embedded or external GUI applications are technical classifications, whereas primary and functional are design classifications.

You must use external GUI applications in a FlowMark workflow process because only external GUI applications can be started by the FlowMark workflow manager. The external GUI applications used in a FlowMark workflow process will follow closely the design rules defined for a primary GUI application as discussed in *Object-Based GUI Application Development with VisualGen*.

1.2 FlowMark

FlowMark helps you automate your business processes. It is designed to handle the flow of work from user to user, following the rules of the business process. Using FlowMark you implement definitions for:¹

- Staff** Staff definitions are used to define skill levels, roles, organizations, relations, and authorizations for staff personnel. The staff definitions map an enterprise to FlowMark staff concepts: people have a level, can belong to an organization, and can play one or more roles. Modelers use these definitions to assign work in process models.
- Servers** Runtime servers associated with process activities are defined as Server objects. Servers can be referenced by process activities.
- Processes** Modelers use the Buildtime Processes folder to define, animate, and maintain process models. Defining a model requires building the workflow diagram with activities and connectors and defining most elements with reference to information contained in the Staff, Data Structures, and Programs folders.
- Data Structures** Data structures associated with programs, processes, or activities are defined as objects. Data structures are used by activities as input or output, and data structure members can be referenced in exit and transition conditions.
- Programs** Programs to be called by program activities or as support tools are registered by specifying the information needed to find and start the program.

For our purposes we are interested in the process and program definitions.

The possible ways in which work and data can flow in a business process are represented graphically by arrows in a workflow model.

A workflow model consists of several elements, including:

- Activities, both program and manual
- Blocks
- Control and data connectors
- Containers for data
- Exit and transition conditions
- Staff

The Buildtime Process diagramming facility is where you specify the business process model. Program activities are connected with flow and data connectors and are associated with a program object, and roles, levels, or staff. Conditions can be specified for exiting an activity or for transitioning from one activity to another.

During runtime, the FlowMark workflow manager resolves which way to go in the business process model. Exit and transition conditions are evaluated for each activity defined. The actual path chosen by the FlowMark workflow manager is dictated by the conditions that are met at runtime.

Figure 1 on page 4 shows a sample process defined in the FlowMark Buildtime Process GUI used to develop process diagrams.

¹ From the IBM FlowMark Online Overview

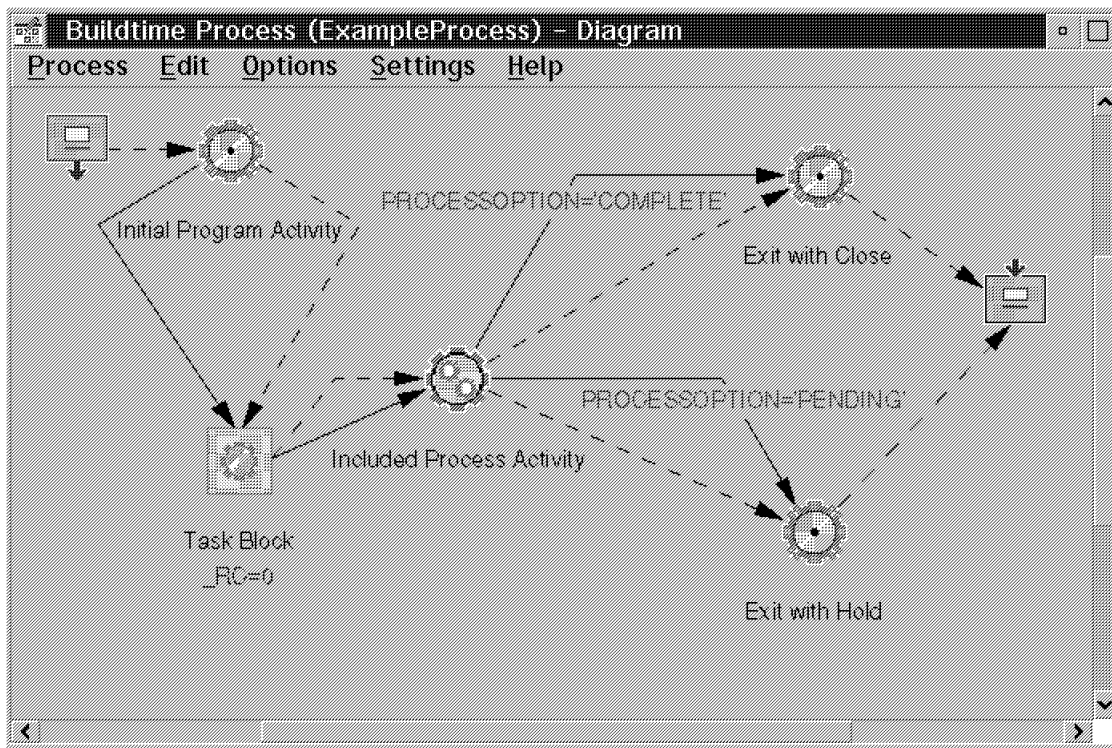


Figure 1. Sample FlowMark Workflow Process Diagram

In Figure 1 there are references to FlowMark data structure members as part of the process definition:

- Exit condition `_RC=0` for the *Task Block* activity
At runtime, the *Task Block* activity will not be complete unless the predefined data structure member, `_RC`, has a value of 0. The predefined data structure member, `_RC`, contains the return code of the program activity.
- Transition conditions `PROCESOPTION='PENDING'` and `PROCESOPTION='COMPLETE'` for the workflow connector from the activity *Included Process Activity* to program activities *Exit with Hold* and *Exit with Close*, respectively.
At runtime, after exiting from the activity *Included Process Activity*, the FlowMark workflow manager launches program activity *Exit with Hold* if `PROCESOPTION='PENDING'` and *Exit with Close* if `PROCESOPTION='COMPLETE'`.

Programs registered in FlowMark are associated with a program activity in a process definition. The program specified in a program registration object is invoked at process runtime by the FlowMark workflow manager. Figure 2 on page 5 shows the settings for the program activity named *Initial Program Activity*.

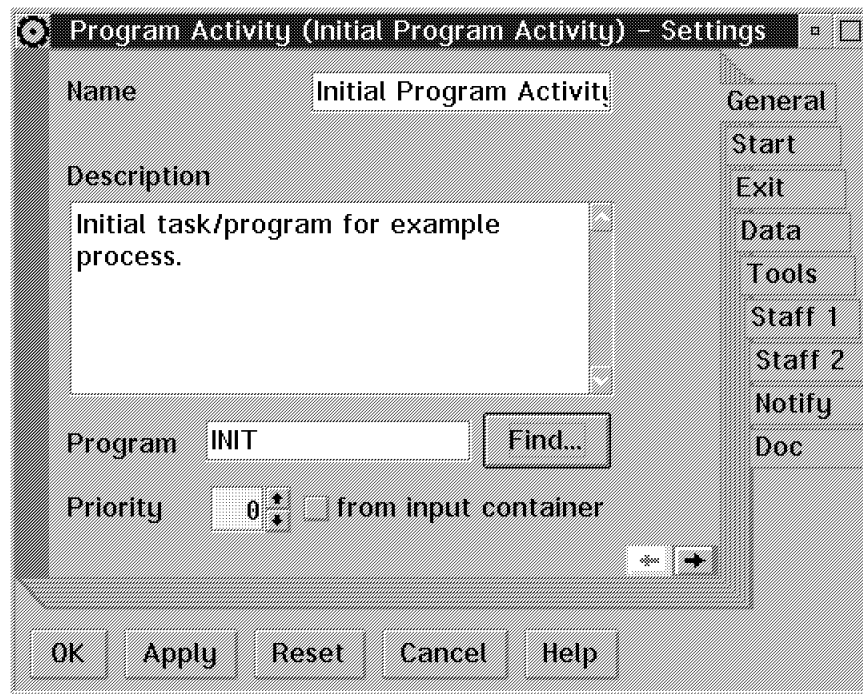


Figure 2. Sample FlowMark Workflow Process Diagram: Initial Program Activity Settings

The program activity identifies the registered program that should be started to implement the process task. Implementation details such as data structures used and operating system specific settings are defined as part of the registered program object (see Figure 3 and Figure 4 on page 6).

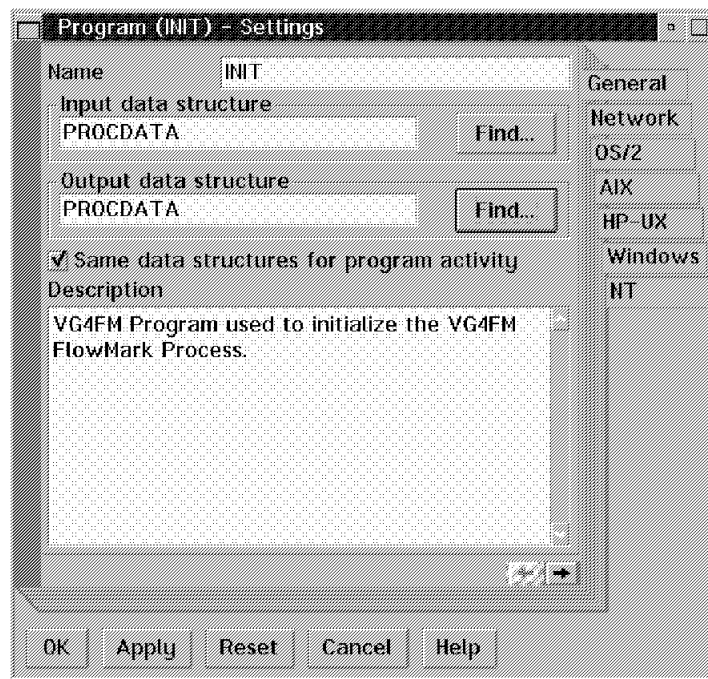


Figure 3. Sample FlowMark Program Registration Definition: INIT Program General Settings

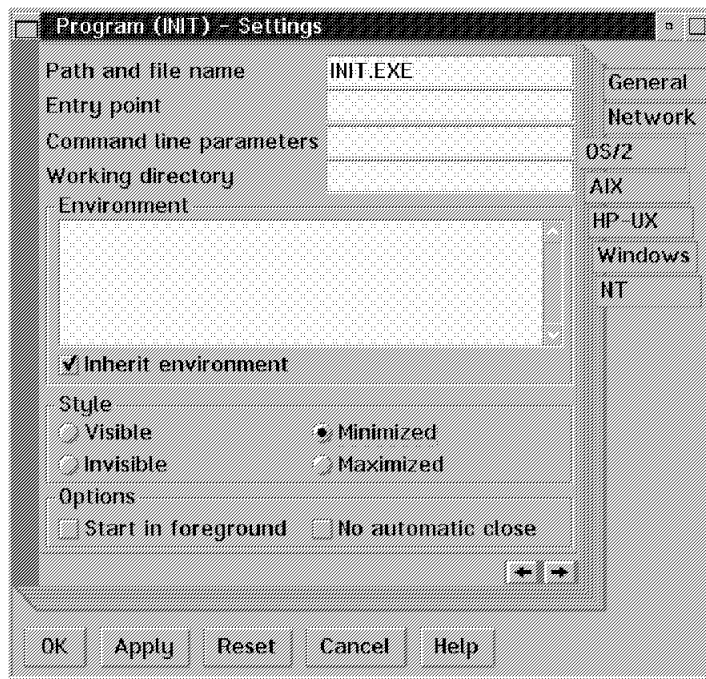


Figure 4. Sample FlowMark Program Registration Definition: INIT Program OS/2 Settings

1.3 FlowMark Workflow Manager

To understand why an interface between FlowMark and VisualAge Generator is necessary, you have to know how the FlowMark workflow manager manages the execution of a program activity.

To log on to the FlowMark runtime, you must have a valid user ID and password. Thus, when you use GUI applications in a FlowMark workflow process, your data is protected from unauthorized access. You can now invoke processes that have been defined in FlowMark and are available in the FlowMark runtime client interface. These processes will need to start program activities that were defined in the process model.

Programs are registered with FlowMark as reusable specifications (see Figure 3 on page 5 and Figure 4). The same program can be associated with multiple program activities. When FlowMark starts a program activity, it launches the program defined in the program registration object for that program activity (see Figure 2 on page 5).

The underlying operating system then creates a session in which the program runs. FlowMark waits until the session ends.

Figure 5 on page 7 shows the FlowMark flow control for ordinary programs.

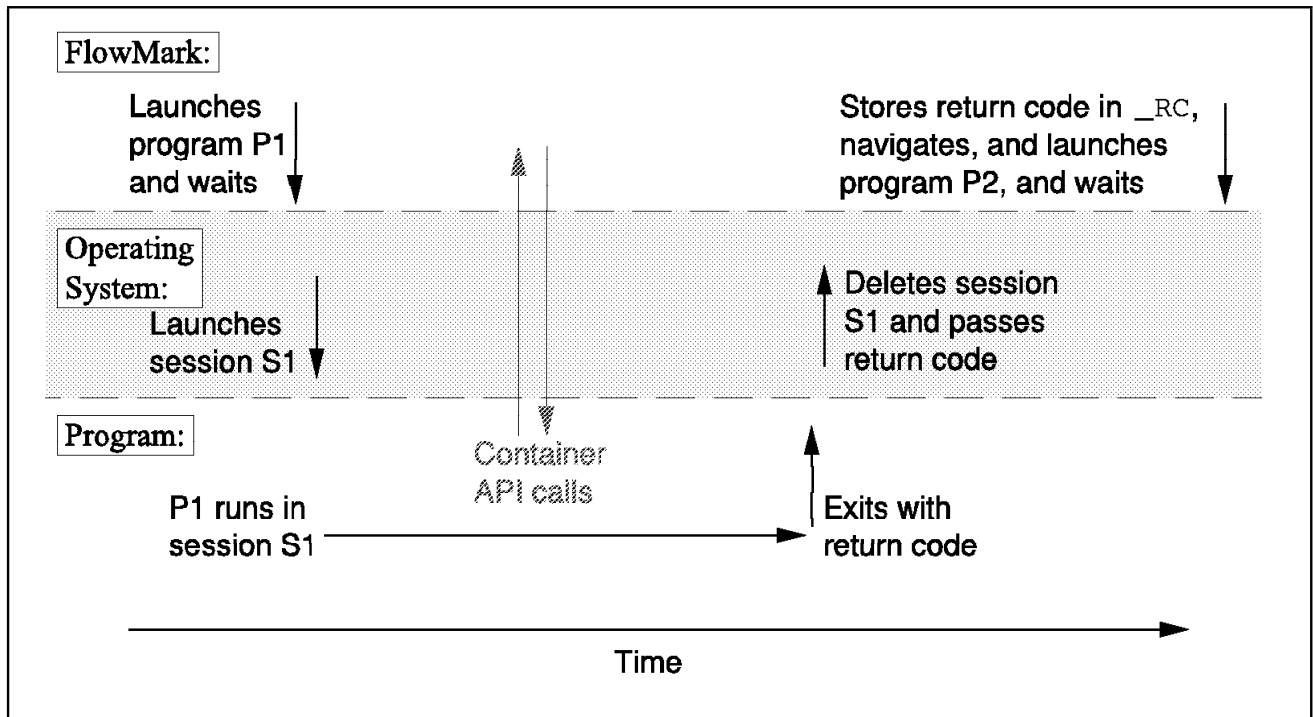


Figure 5. FlowMark Flow Control for Programs

The sequence and layering shown in Figure 5 can be described as:

- Start FlowMark program activity
- Start operating system session
 - Start program
 - Wait while program function is performed
 - Program ends
- End operating system session
- End FlowMark program activity

When the program is finished, it returns a return code. (If no value is assigned to the return code, its default value is 0.) The operating system deletes the session and passes the return code back to FlowMark.

The return code is then saved in the FlowMark output container in the predefined data structure member, `_RC`. The program might have saved values in other members of the FlowMark output container through FlowMark application programming interface (API) calls.

On the basis of the values of the data structure members in the program activity output container, FlowMark checks the exit condition and the transit condition, determines which way to go in the process path, and launches the next program activity.

1.4 VisualAge Generator Integration Issues

A VisualAge Generator GUI runs in the VisualAge Generator runtime environment, which is a Smalltalk environment. Therefore, when a VisualAge Generator GUI command file starts a GUI application, the VisualAge Generator runtime environment is automatically started, if it is not already running.

When the command file launches the GUI application, it finishes and returns 0 as a return code to the operating system, which passes it back to FlowMark.

Figure 6 shows the content of the command file for the VisualAge Generator GUI *VGGUI*. The VisualAge Generator runtime environment is invoked by the EZE2RUN command.

```
@EZE2RUN OPEN VGGUI
```

Figure 6. Content of Command File for VisualAge Generator GUI VGGUI

The VisualAge Generator GUI command file does not wait until the GUI application is finished; instead, it runs for a very short time and finishes *before* the GUI application finishes.

When the GUI application finishes, the VisualAge Generator runtime environment continues to run. When a GUI application starts the VisualAge Generator runtime environment, the operating system will start the VisualAge Generator runtime environment in a separate session. The VisualAge Generator runtime environment session will run until a GUI runtime stop (EZE2RUN STOP) command is issued.

Figure 7 shows the FlowMark flow control for a VisualAge Generator GUI application.

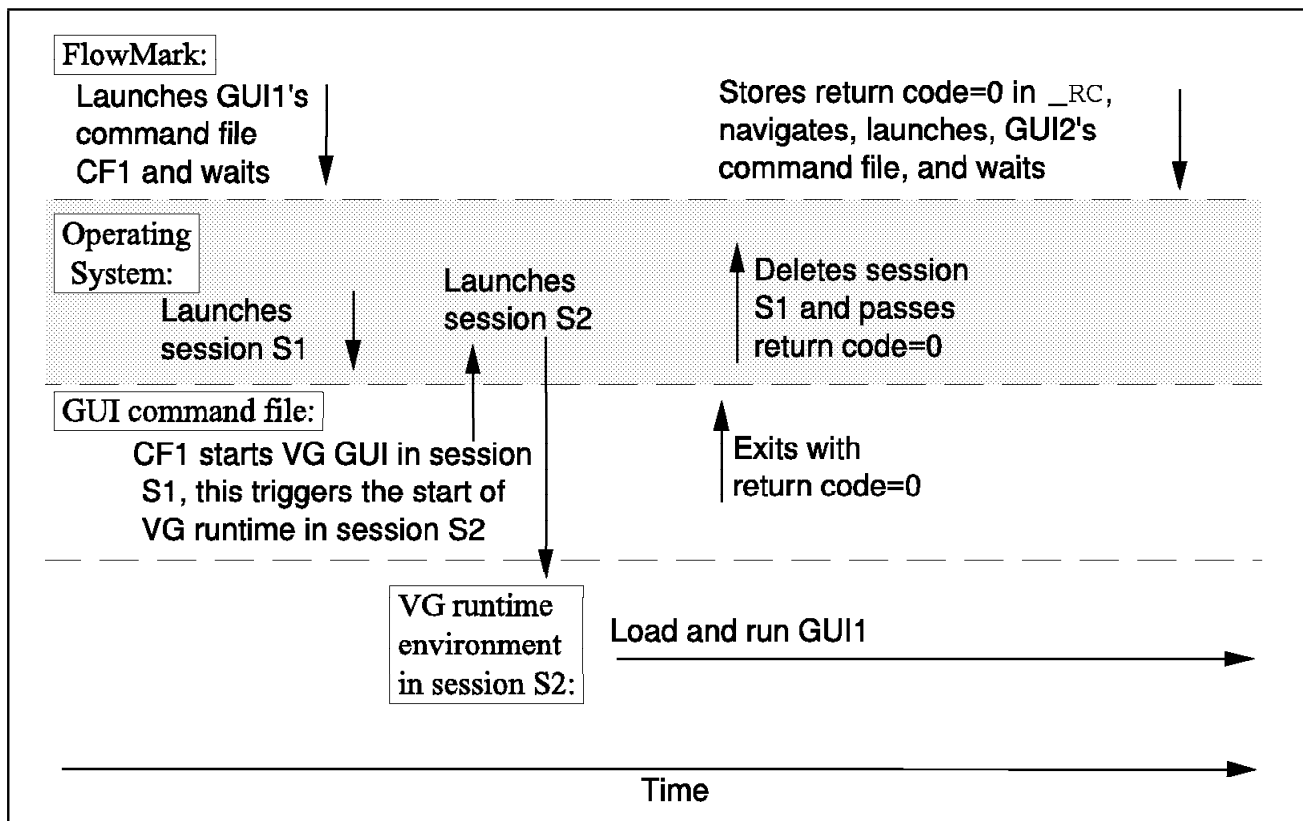


Figure 7. FlowMark Flow Control for VisualAge Generator GUI Application

For all GUI applications that FlowMark starts, it receives a return code of 0 from the VisualAge Generator GUI command files before the GUI application finishes. If you have adjacent GUI applications in a FlowMark workflow process, all GUI applications are started before the previous GUI application (as defined in the FlowMark workflow process) has performed a real task (that is, the task for which the GUI application was developed).

The FlowMark workflow manager decides which path to enter in the process *before* GUI1 is finished and starts GUI2. As a consequence GUI2 may end abnormally if it depends on data from GUI1.

Thus it is impossible for the FlowMark workflow manager to control when a GUI application finishes and use _RC for exit or transition conditions. The FlowMark workflow manager therefore cannot navigate a process path if a VisualAge Generator GUI is used as a program activity in the process.²

Because VisualAge Generator GUIs run in a Smalltalk runtime environment, you cannot pass command line parameters to a VisualAge Generator GUI application. Therefore it is difficult to create an interface, as we explain in 2.1, "FlowMark Integration" on page 11.

One factor that influences the implementation of the interface program is that shared memory allocated or files created from a GUI application are locked by the VisualAge Generator runtime environment until the GUI application is completely closed.

² The same problem exists for GUI applications developed with VisualAge Smalltalk. There are integration approaches where VisualAge Smalltalk GUI applications using broker/requestor parts and FlowMark broker/requestor interfaces can be used in a FlowMark workflow process. VisualAge Generator Version 2.2 GUI application development does not provide support for these broker/requestor parts.

Chapter 2. Integrating VisualAge Generator with FlowMark

In this chapter we identify the integration objectives for using VisualAge Generator GUIs in FlowMark processes and describe two options for implementing a FlowMark-to-VisualAge Generator interface. Each option uses a different method for communicating between the interface program and the hidden GUI application.

2.1 FlowMark Integration

In this section we discuss the objectives and options for a FlowMark-to-VisualAge Generator interface.

2.1.1 Objectives

As discussed in 1.3, “FlowMark Workflow Manager” on page 6, a program effectively integrated as a FlowMark program activity process task managed by the FlowMark workflow manager has these attributes:

Task/Program Start Linked

Start of FlowMark program activity process task is linked with the start of program execution.

FlowMark Control Information Available

FlowMark control information is available to the started program through the use of the appropriate FlowMark API calls by the program.

Task/Program End Linked

End of program execution is linked with the end of the FlowMark program activity process task. The FlowMark program activity process task does not end until the program ends.

Program Return Code Available

Program return code information is available to FlowMark as control information to support process navigation.

A program activity process task can be in these states:

- | | |
|-----------------|---|
| Starting | The FlowMark workflow manager is in control. An instance of the program specified in the program registration object associated with the program activity is being created and started. |
| Running | FlowMark control information is available to the started program through the use of the appropriate FlowMark API calls by the program. |
| Ending | End of program instance execution is linked with the end of the FlowMark program activity. The FlowMark program activity does not end until the program instance ends. |

Ended Program return code information is available to the FlowMark workflow manager as control information to support process navigation.

As discussed in 1.4, “VisualAge Generator Integration Issues” on page 7, there are problems related to using VisualAge Generator GUI applications as FlowMark program activities. To integrate VisualAge Generator GUI applications with FlowMark we must be able to match the identified FlowMark program activity process attributes and process states.

2.1.2 Options

Now that you know that the FlowMark workflow manager cannot control a GUI application, what can you do? You simply hide the GUI application from FlowMark!

The basic structure for hiding the GUI application is to define an interface program that will be started when the FlowMark program activity is started. The relationship of the interface program to the FlowMark program activity and the VisualAge Generator GUI application is outlined below:

- Start FlowMark program activity
- Start operating system session
 - Start VisualAge Generator-FlowMark interface program
 - Start VisualAge Generator GUI application
 - Wait while GUI application function is performed
 - End VisualAge Generator GUI application
 - End VisualAge Generator-FlowMark interface program
- End operating system session
- End FlowMark program activity

The solution requires that you hide the GUI application behind an ordinary program. You write an interface program, which can be controlled by the FlowMark workflow manager. The interface program launches, communicates with, and controls the termination of the GUI application and at the same time communicates with FlowMark through API calls.

Your solution must not use FlowMark containers as a database for the GUI applications (see Chapter 10, “Data Container Usage” in the *Image and Workflow Library: FlowMark V2.2 Design Guidelines*). You must keep application data in the application databases.

When you use GUI applications in a FlowMark workflow process, you can use only external GUI applications, because only external GUI applications can be launched from the FlowMark workflow manager or any other program. Therefore, you will have more external GUI applications than you would for a solution that does not use FlowMark. The result is that you need more external communication among the external GUI applications. In a solution that does not use FlowMark, the communication is internal between an external GUI application and its subsidiary embedded GUI applications. (See Appendix A, “GUI Application Terms and Implications” on page 45 for descriptions of embedded and external GUI applications.)

Many instances of a FlowMark workflow process can run concurrently. You can have the interface program handle all FlowMark workflow process instances, or you can have one instance of the interface program for every FlowMark workflow process instance. The simplest solution is to make one instance of the interface program correspond to one instance of a FlowMark workflow process.

So how does a GUI application know by which FlowMark workflow process it was invoked? That is a little difficult to tell because you cannot pass the unique FlowMark

identification as a command line parameter to the VisualAge Generator GUI. Implementation depends on which communication method you choose.

A GUI application can communicate with the interface program, or any other non-GUI application, through either a dynamic link library (DLL) or dynamic data exchange (DDE).

2.2 Communication Using a Dynamic Link Library

When you use the DLL approach, the interface consists of two components: an EXE program, which the FlowMark workflow manager can control, and a DLL with which both the GUI application and the EXE program under the control of the FlowMark workflow manager can communicate. Hereafter, we call the EXE part of the interface program *STARTGUI.EXE* and the DLL part of the interface program *VG4FMDLL.DLL*.

Figure 8 shows how the FlowMark workflow manager communicates with the EXE part of the interface program and the GUI application communicates with the DLL part of the interface program.

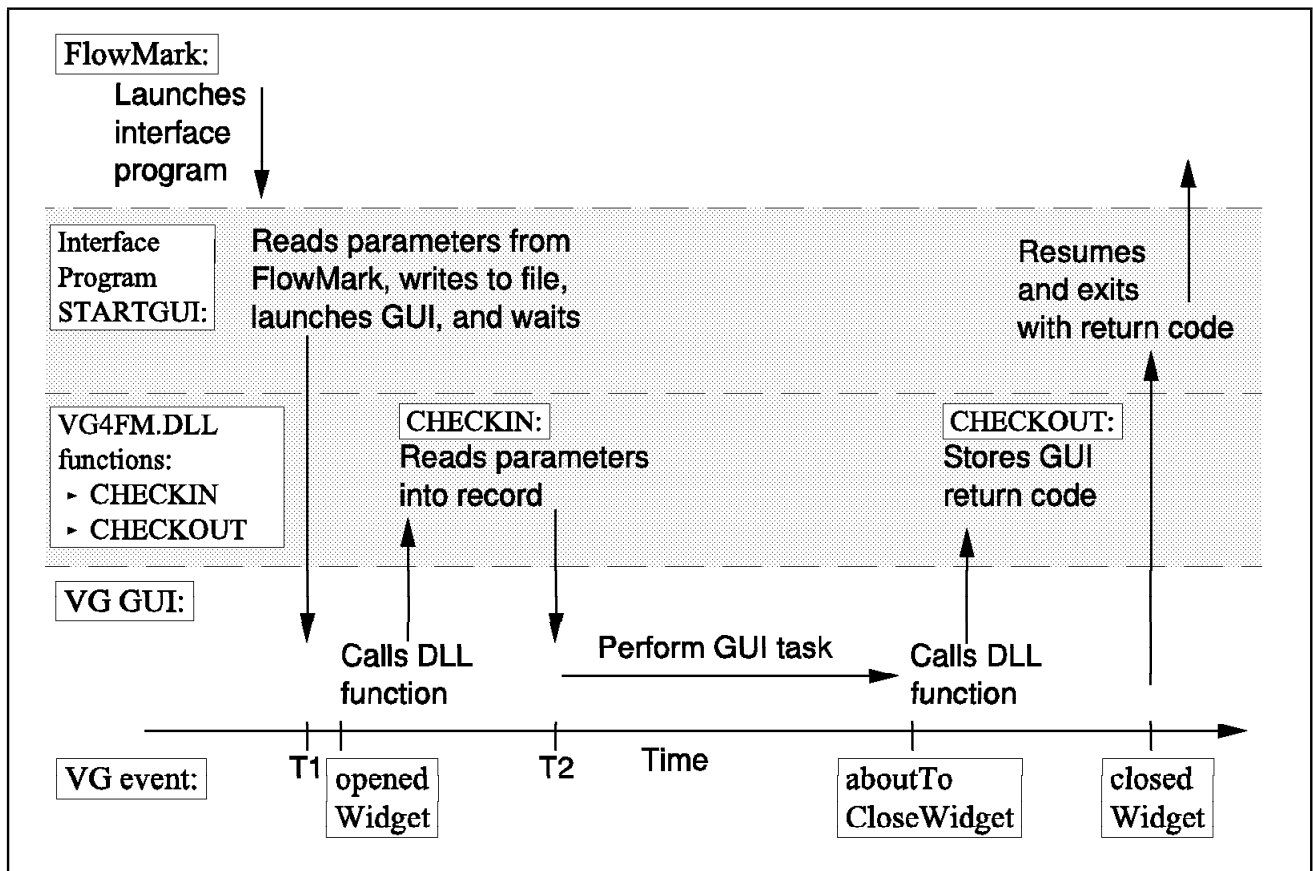


Figure 8. DLL Communication between the VisualAge Generator GUI and FlowMark through the Interface Program

2.2.1 STARTGUI.EXE

STARTGUI.EXE launches the GUI application. As the GUI application cannot receive any parameters from the launching STARTGUI.EXE, they must be written into a file or shared memory from which the GUI application can read them.

After the FlowMark workflow manager has launched STARTGUI.EXE, STARTGUI.EXE reads the unique FlowMark instance ID and other parameters (such as user ID) through API calls. STARTGUI.EXE writes the parameters into a file or shared memory and then launches the GUI application.

After launching the GUI application, STARTGUI.EXE waits until the GUI application closes. STARTGUI.EXE can then close and eventually return a return code to FlowMark. The FlowMark workflow manager can then navigate the process path and start the next program activity.

To detect when the GUI application has closed, either normally or abnormally (that is, with a trap), STARTGUI.EXE must periodically check whether or not the VisualAge Generator GUI is a valid window. Therefore, STARTGUI.EXE in OS/2 must be a Presentation Manager (PM) program and retrieve the window handle of the VisualAge Generator GUI. A similar requirement exists for implementation on Windows NT.

2.2.2 VG4FMDLL.DLL

In order for the GUI application to be independent of how the external communication is implemented, VG4FMDLL.DLL should handle the external communication. The GUI application must check in and out of VG4FMDLL.DLL, to say "Now I am here" and "Now I am gone," as we check in and out of a hotel. We have therefore named the two functions in VG4FMDLL.DLL *CHECKIN* and *CHECKOUT*.

VG4FMDLL.DLL and the GUI application communicate through a record that is declared in both VG4FMDLL.DLL and the GUI application. The GUI application always uses a working storage record for communication, no matter how the external communication is implemented in VG4FMDLL.DLL. This simplifies maintenance of the GUI applications.

2.2.3 GUI Application

When the GUI application opens (Event: *openedWidget*), it has to read the parameters to know to which process instance it corresponds. So the GUI application, in response to the *openedWidget* event, calls the *CHECKIN* function of VG4FMDLL.DLL.

Some of the parameters obtained during the call to the *CHECKIN* function of VG4FMDLL.DLL can be used to control the appearance of the VisualAge Generator GUI—for example, disable buttons. Others can tell the GUI application where to read and write data for external communication.

After the GUI application has read the parameters, it performs its GUI task. Then, just before the GUI application terminates (Event: *aboutToCloseWidget*), it calls the VG4FMDLL.DLL *CHECKOUT* function, which writes the GUI application's return code (and other values) into a FlowMark user-defined data structure member. In the same function call, the GUI application can write external data to be used by the next GUI application in the workflow process.

The GUI application can now close and STARTGUI.EXE can resume. Once STARTGUI.EXE has closed, the FlowMark workflow manager can navigate the process path and start the next program activity.

2.2.4 File-Based Communication

We chose to use a file to support the external communication required for our interface program. The advantage of choosing to use a file instead of shared memory is that you can read, create, and delete a file from the command line. This is very helpful when you are developing the interface program or the GUI applications. Shared memory is slightly faster. However, compared with the time it takes to load a GUI application, it does not matter whether you choose a file or shared memory. If you want your solution to run in Windows V3.11 and OS/2 or Windows V3.11 and Windows NT, you can reuse most of the code if you use files.

For VG4FMDLL.DLL to read the parameters from the file or shared memory, it must know the name of the file or shared memory. Because the GUI application cannot get the unique name from STARTGUI.EXE, which launched it, it cannot pass a unique name to the CHECKIN function. The name of the file or shared memory must be *hard-coded* in STARTGUI.EXE and the CHECKIN function. Therefore all GUI applications must use the same file or shared memory.

To ensure that the GUI applications do not mix parameters, only one instance of STARTGUI.EXE is allowed to write to the file or shared memory at a time. Until the corresponding GUI application has read the parameters through the CHECKIN function, all other instances of STARTGUI.EXE must wait to write their parameters to the file or shared memory. Thus, from time T1 to T2, no other GUI application can be launched. Time T1 is the time when the file or shared memory is created, and time T2 is the time when the file or shared memory is read.

This is not a serious constraint because two GUI applications are seldom launched at the exact same time on one machine. Should that occur, one instance of STARTGUI.EXE would wait less than a tenth of a second before it launches its GUI application.

In Chapter 3, "Implementing a FlowMark-to-VisualAge Generator Interface" on page 19 we discuss the implementation of a DLL-based interface for VisualAge Generator GUI applications in a FlowMark system.

2.3 Communication Using Dynamic Data Exchange

When you use DDE, the interface consists of only one component: an EXE program, which the FlowMark workflow manager can control. Hereafter we call the EXE program *STARTDDE.EXE*. STARTDDE.EXE is the DDE server, and the GUI application is the DDE client. If you use OS/2, STARTDDE.EXE must be a PM program, as only PM programs can do DDE communication.

Figure 9 on page 16 shows how a GUI application communicates with FlowMark through the interface program, using a DDE conversation. The interface program communicates with FlowMark through API calls.

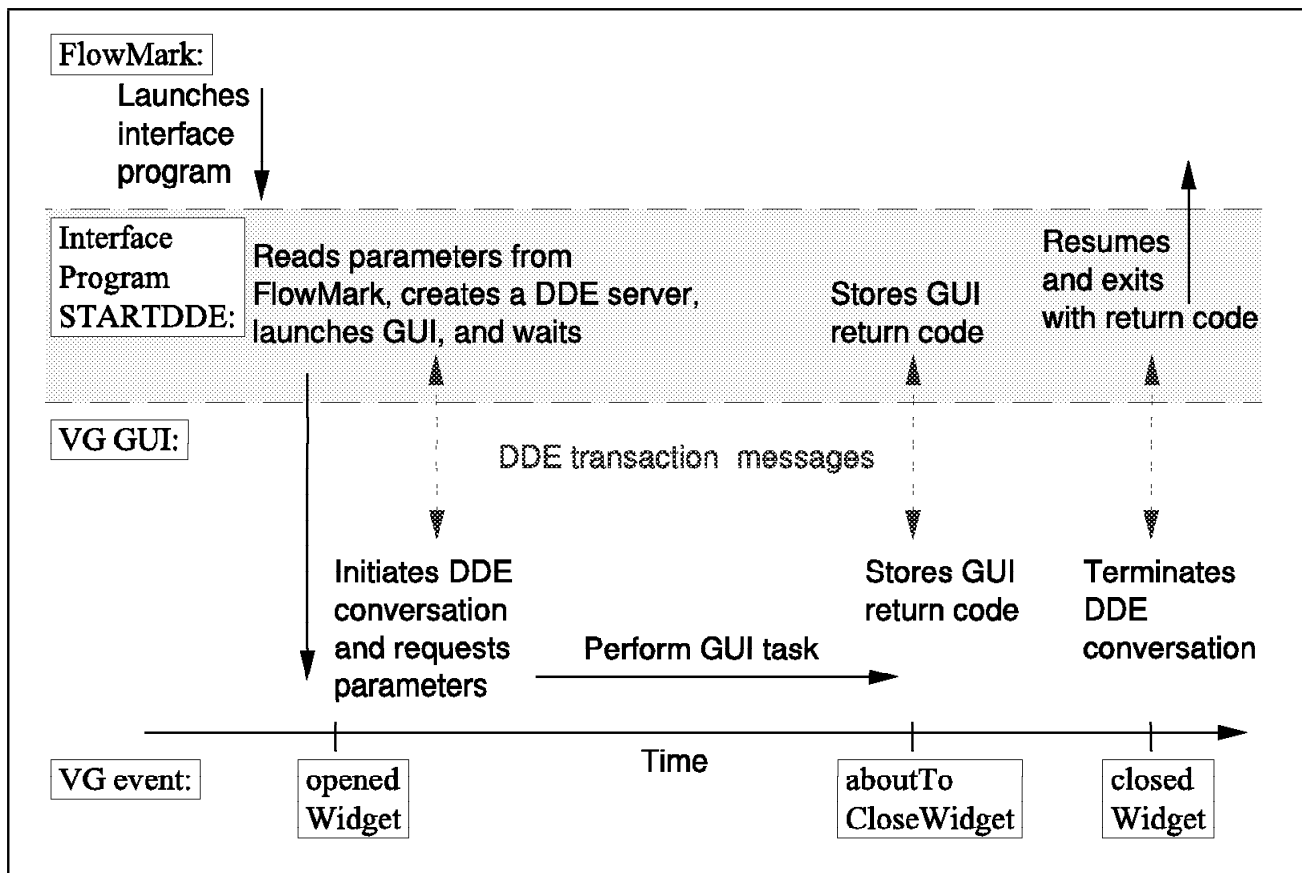


Figure 9. DDE Communication between the VisualAge Generator GUI and FlowMark through the Interface Program

2.3.1 STARTDDE.EXE

STARTDDE.EXE launches the GUI application. Because the GUI application cannot receive any parameters from the launching STARTDDE.EXE, each GUI application requests the parameters from STARTDDE.EXE, using a DDE conversation.

After the FlowMark workflow manager has launched STARTDDE.EXE, STARTDDE.EXE reads the unique FlowMark instance ID and other parameters (such as user ID) through API calls. STARTDDE.EXE then creates a DDE server, launches the GUI application, and waits for the GUI application to initiate the DDE conversation. When the DDE server is created, STARTDDE.EXE is specified as the server application name, and the name of the GUI application is specified as the topic name. STARTDDE.EXE can read the unique FlowMark instance ID after the DDE conversation is established. STARTDDE.EXE has only to retrieve the FlowMark instance ID before it posts the FlowMark instance ID to the GUI application.

To detect when the GUI application has closed, either normally or abnormally, STARTDDE.EXE must periodically check whether or not the VisualAge Generator GUI is a valid window (exactly as for the DLL approach). But the DDE server gets the window handle of the VisualAge Generator GUI when the GUI application initiates the DDE conversation, so in this case there is no need to retrieve the window handle.

2.3.2 GUI Application

When the GUI application opens (Event: `openedWidget`), it must read the parameters to know to which FlowMark workflow process instance it corresponds. Some parameters can be used to control the appearance of the VisualAge Generator GUI—for example, disable buttons. Others can tell the GUI application where to read and write data for external communication. The GUI application (the DDE client) initiates the DDE conversation by posting a message (`WM_DDE_INITIATE`) to the DDE server. To initiate the DDE conversation, the GUI application must have the server application name and the topic name. All GUI applications must use the application name `STARTDDE.EXE` and the name of the GUI application itself as the topic name when the DDE conversation is sent. You must *hard-code* these names into the GUI application as they cannot be passed to the GUI application.

2.3.3 DDE-Based Communication

To ensure that the GUI applications do not mix parameters or data, only one DDE conversation is allowed for each server. Thus, when a DDE server has posted a positive acknowledgment to a GUI application, the DDE server must post negative acknowledgments to all other incoming DDE conversation requests. Thus, the DDE server posts a positive acknowledgment to only those DDE conversation requests that contain `STARTDDE.EXE` as the server name *and* the name of the GUI application as the topic name. The DDE server must post negative acknowledgments to all other incoming DDE conversation requests.

If three instances of `STARTDDE.EXE` are started at the same time by the same program activity, three identical DDE servers are created, and three identical GUI applications each initiate one DDE conversation. The DDE conversations are established on a first-come first-serve basis. However, the GUI applications are identical, so it does not matter which GUI application connects to which instance of `STARTDDE.EXE`. If the DDE servers are created by instances of `STARTDDE.EXE` launching different GUI applications, the DDE servers post a positive acknowledgment only to the GUI application with the same topic name, so that there is no mixing of data or parameters.

After the GUI application has requested and received the parameters from the DDE server, it can perform its real task. In order for the GUI application to be independent of the external communication implementation, `STARTDDE.EXE` implements the external communication. `STARTDDE.EXE` and the GUI application communicate through DDE transactions. The GUI application always uses the DDE transactions for communication no matter how the external communication is implemented in `STARTDDE.EXE`.

When the GUI application has accomplished its real task (Event: `aboutToCloseWidget`), it can then post its return code to `STARTDDE.EXE`, which writes the GUI application's return code (and other values) into a FlowMark user-defined data structure member. The GUI application can also post data to be used by the next GUI application in the workflow process.

The GUI application ends the DDE conversation by posting a message (`WM_DDE_TERMINATE`) and closes itself (Event: `closedWidget`). The waiting `STARTDDE.EXE` can then resume. When `STARTDDE.EXE` terminates, the FlowMark workflow manager can navigate the process path and start the next program activity.

Compared with DLL-based communication, the DDE conversation takes more system resources at runtime because the DDE server has to check all incoming conversation requests and post a positive or negative acknowledgment.

We did not implement a DDE-based interface during the residency.

Chapter 3. Implementing a FlowMark-to-VisualAge Generator Interface

In this chapter, we describe how to implement the interface program when you use the DLL approach discussed in 2.2, "Communication Using a Dynamic Link Library" on page 13. The description covers both the OS/2 and Windows NT runtime environments.

All FlowMark, VisualAge Generator, and interface program materials are included on the diskette accompanying this redbook. When you look at the interface code samples, you can see that the solution for OS/2 and Windows NT is almost identical.

3.1 FlowMark Workflow Manager Definitions

The first thing you need is a FlowMark workflow process that supports the invocation of a VisualAge Generator GUI application using the interface program. Several steps are required in the FlowMark workflow process that supports GUI application invocation: initialization, invocation, and cleanup/exit. Figure 10 shows the VG4FMGUI FlowMark workflow process diagram that includes program activities for each of these steps.

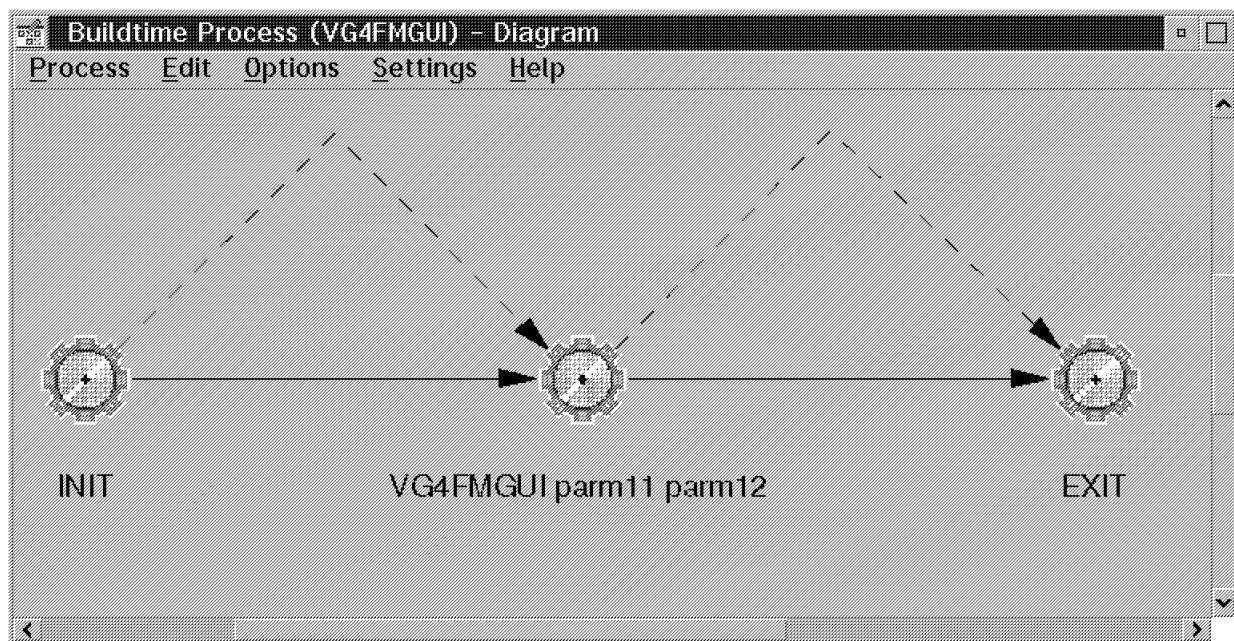


Figure 10. VG4FMGUI FlowMark Workflow Process Diagram

In Figure 10 the solid lines represent process control flows, and the dashed lines represent data flows. The FlowMark workflow process defined uses a user-defined

data structure to contain information required to support the process and the VisualAge Generator-to-FlowMark interface program. The program activities shown in Figure 10 transfer data using a user-defined data structure named VG4FM (see Figure 11 on page 20). FlowMark also provides predefined data structure members such as `_RC` and `_ACTIVITY`, which are included in any user-defined data structure member.

Name	Type
-----	-----
FILENAME	STRING
DESCRIPTION	STRING
RETURNCODE	STRING

Figure 11. FlowMark Definition of VG4FM Data Structure

The INIT program activity is responsible for initializing user-defined data structure members `FILENAME` and `DESCRIPTION`. The program `INIT.EXE` is assigned to program activity `INIT`. When `INIT.EXE` ends, the next program activity starts. (The `INIT.EXE` program is discussed in 3.2, “FlowMark-VisualAge Generator Initialization Program” on page 22.)

The *VG4FMGUI parm11 parm12* program activity shown in Figure 10 on page 19 is responsible for launching the GUI application. The `STARTGUI.EXE` program is assigned to program activity *VG4FMGUI parm11 parm12*. (The `STARTGUI.EXE` program is discussed in 3.3, “FlowMark-VisualAge Generator Interface EXE Program” on page 22.)

When the GUI application terminates, program activity `EXIT` is started. Program `EXIT.EXE` is assigned to program activity `EXIT`. `EXIT.EXE` deletes all files created during the current instance of the process.

If the GUI application started by `STARTGUI.EXE` ends with an abnormal return code, the FlowMark workflow process suspends the current activity program. When the activity program is resumed, it should restart the GUI application. With the FlowMark workflow process shown in Figure 10 on page 19, the GUI application would not restart; the process would just run to completion.

We found that if we replaced the `STARTGUI.EXE` activity program with a task block and implemented the task block with an activity program that used exit criteria, we could get the results we wanted. When the GUI application ends with an abnormal return code, the FlowMark workflow process suspends, and when resumed, the GUI application is restarted.

Figure 12 on page 21 shows the FlowMark process diagram and task block exit criteria (`RETURNCODE='OK'`) for process *VG4FMGUI-Blocked*.

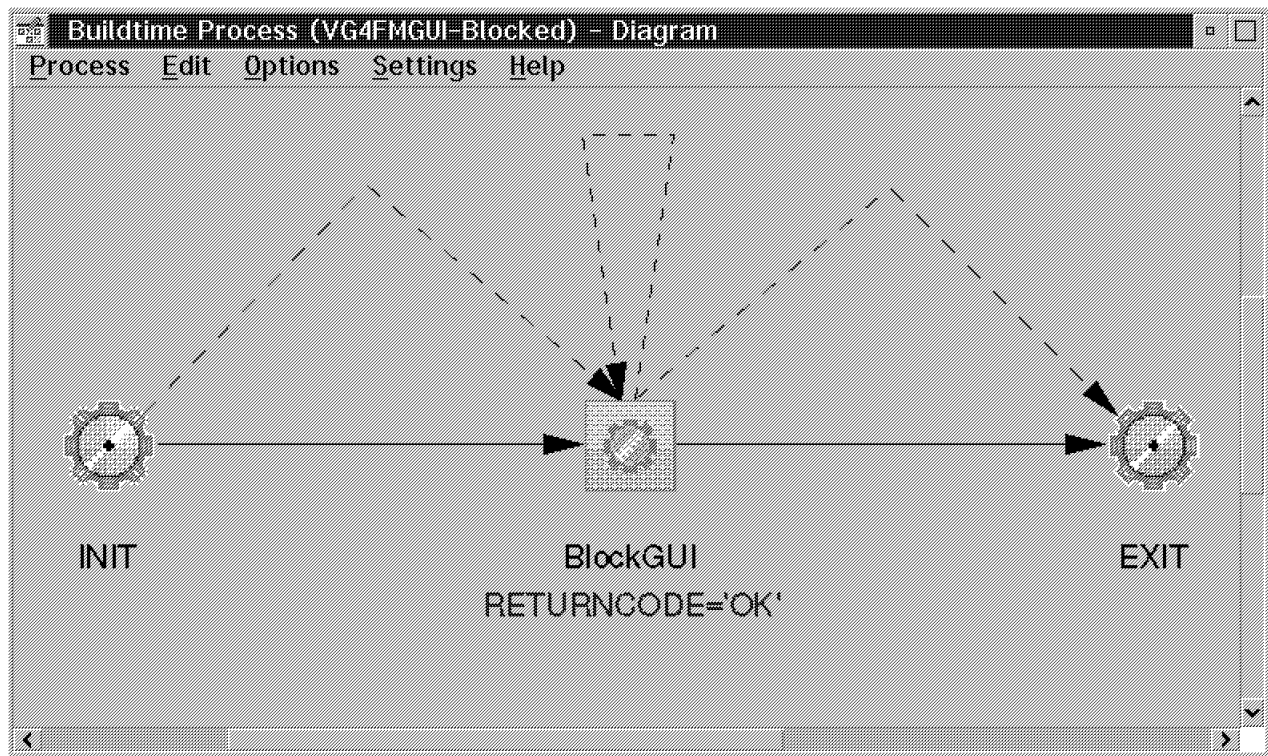


Figure 12. VG4FMGUI Blocked FlowMark Workflow Process Diagram

Figure 13 shows the FlowMark process diagram for the *BlockGUI* task block.

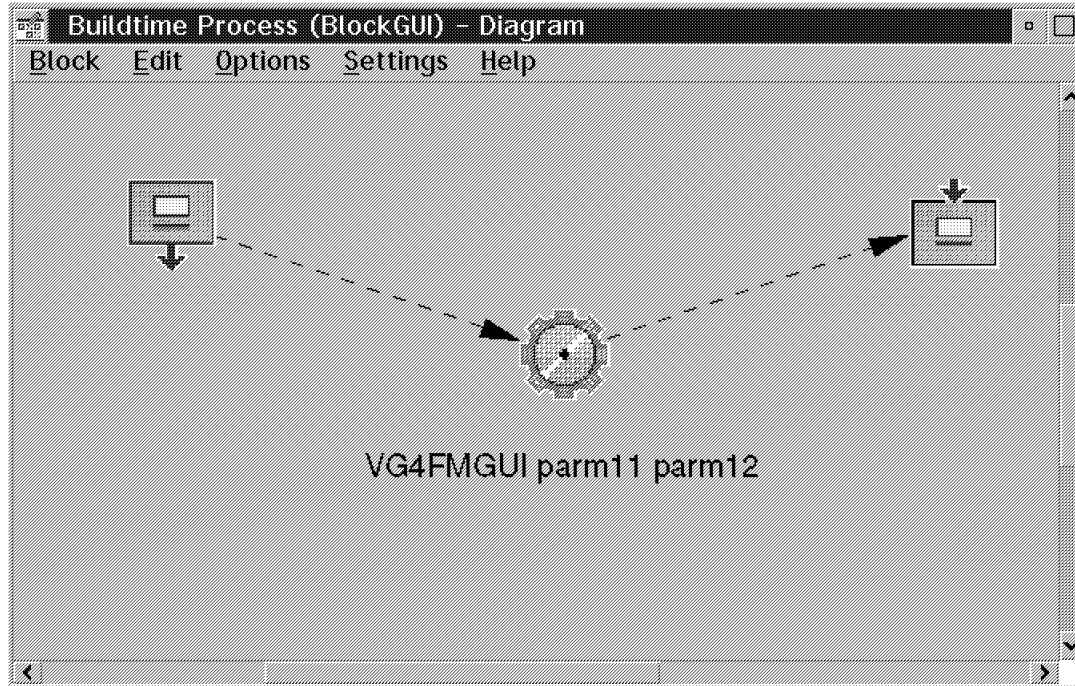


Figure 13. BlockGUI FlowMark Workflow Process Diagram

We review the use of the *VG4FMGUI-Blocked* process in Chapter 4, “VisualAge Generator and FlowMark Sample Application” on page 33.

All of the FlowMark definitions we use in our sample are provided in the VG4FMPC.FDL file, which is on the diskette included with this redbook.

3.2 FlowMark-VisualAge Generator Initialization Program

We wrote a program, INIT.EXE, that stores the unique file name in the FlowMark user-defined data structure member, FILENAME. Subsequent program activities in the current instance retrieve the file name from FlowMark through API calls, thus ensuring that all program activities in the current instance use the same file name. Therefore each FlowMark workflow process must start with a program activity that starts INIT.EXE.

Figure 14 shows the basic logic of the INIT.EXE program in pseudo-code.

```
Get unique current FlowMark process instance ID

Create unique filename from process instance ID and save in FM container

Set FM container member DESCRIPTION to blanks
```

Figure 14. Initialization Program INIT.EXE Pseudo-Code Logic

INIT.EXE also sets the value of the FlowMark user-defined data structure member, DESCRIPTION, to blanks (space characters). DESCRIPTION is not necessary but is a convenience for the end user. If %DESCRIPTION% is specified in the description field of a program activity, the end user will see the actual value of the user-defined data structure member, DESCRIPTION, under the instance icon in the Work List folder.

When end users run several concurrent instances of the same FlowMark workflow process, they can distinguish instances by looking at the actual value of the user-defined data structure member, DESCRIPTION. The GUI application sets the value of DESCRIPTION through the DLL CHECKOUT function.

A typical value of DESCRIPTION would be a customer name or social security number. The user-defined data structure member, RETURNCODE, is also set by the GUI application and is used instead of the predefined data structure member, _RC.

The GUI application typically also needs the end user's FlowMark user ID, which STARTGUI.EXE retrieves through a FlowMark API call. Thus GUI applications can use the security system built into FlowMark instead of implementing a new system.

3.3 FlowMark-VisualAge Generator Interface EXE Program

From reading Chapter 2, "Integrating VisualAge Generator with FlowMark," you know that, to avoid mixing parameters, before launching its GUI application the current instance of STARTGUI.EXE must wait until other instances of STARTGUI.EXE have launched their GUI applications.

This requires that a synchronization mechanism be chosen. This mechanism can be a semaphore, a shared memory object, or a file, and code that tests whether the semaphore is set, or the shared memory object is available, or the file exists. Because we needed to pass parameters from STARTGUI.EXE to VG4FMDLL.DLL, we did not want to use the semaphore. We chose to use a file because then we did not have to use API calls that are operating system dependent, and we did not have to write special programs to create, delete, or view the content of the shared memory object (we kept it simple).

Figure 15 on page 23 shows the STARTGUI.EXE pseudo-code logic. The C:\STRT_GUI.FIL file is used by all instances of STARTGUI.EXE to pass parameters to the GUI application.

```
Cleanup after any previous instance of STARTGUI.EXE

Wait until no other GUI application is launched

Retrieve parameters and save them in C:\STRT_GUI.FIL

Launch the GUI application

Wait until the GUI application terminates

If the GUI application terminated abnormally set return code accordingly

Exit current instance of STARTGUI.EXE with return code set
```

Figure 15. Interface Program STARTGUI.EXE Pseudo-Code Logic

The cleanup is done first because the VisualAge Generator runtime environment locks any resources such as files and shared memory used by the GUI application until the GUI application terminates. When the FlowMark workflow manager starts a GUI application, we know for sure that the previous GUI application has terminated and we will not have problems removing files or shared memory.

Before STARTGUI.EXE launches the GUI application, it retrieves parameters for the GUI application from FlowMark through API calls. The GUI application needs at least an identification of the FlowMark workflow process instance that invokes the GUI application, to establish communication with a previous or future GUI application in the same FlowMark workflow process instance. For example, the GUI application could read keys written by a previous GUI application that will be used to look up values in a database. The place to read and write such keys could be a file or shared memory. In this redbook and in the code samples, we chose to show only the file approach, because file handling in C is simple and is independent of the operating system. The file name must be unique to prevent mixing data among FlowMark workflow process instances.

We chose to use the unique FlowMark workflow process instance name as a part of the file name. This simple measure makes it possible for you to track data flow easily during the instance. If the name of the FlowMark workflow process instance is VG4FMGUI_61, all GUI applications in this instance could use the unique file C:\FMGUI_61.DAT for storing key values. This approach is implemented in the code samples.

STARTGUI.EXE can pass other parameters to the GUI application—for example, the name of the program activity that can be retrieved from the predefined data structure member, `_ACTIVITY`. In our implementation of STARTGUI.EXE, you see that the first three words of the name of the program activity are passed to the GUI application as three parameters. If the name of the program activity is VG4FMGUI NO EDIT, the first parameter is VG4FMGUI, the second parameter is NO, and the third parameter is EDIT. The parameters can be used to disable entry fields or buttons in the GUI application, thereby making it possible to use the same GUI application for multiple purposes, such as providing read-only or update access to business data.

The name of the GUI application to be launched is passed as a parameter to STARTGUI.EXE. When you create the GUI application program registration object, you specify the path and file name as STARTGUI.EXE and the name of the GUI application as a command line parameter. Specify that STARTGUI.EXE starts invisibly. The end user wants to see the GUI application window, not the STARTGUI.EXE window, which is empty all the time. If you specify the setting on both the OS/2 attribute page and the Windows NT attribute page, the same FlowMark workflow process runs on both operating systems. You do not need any other modifications. Even the generated VisualAge Generator GUI application file, VG4FMGUI.APP, is identical for OS/2 and Windows NT. VG4FMGUI.APP is on the diskette accompanying this redbook.

Figure 16 on page 24 shows the general settings for the FlowMark program registration object for GUI application VG4FMGUI.

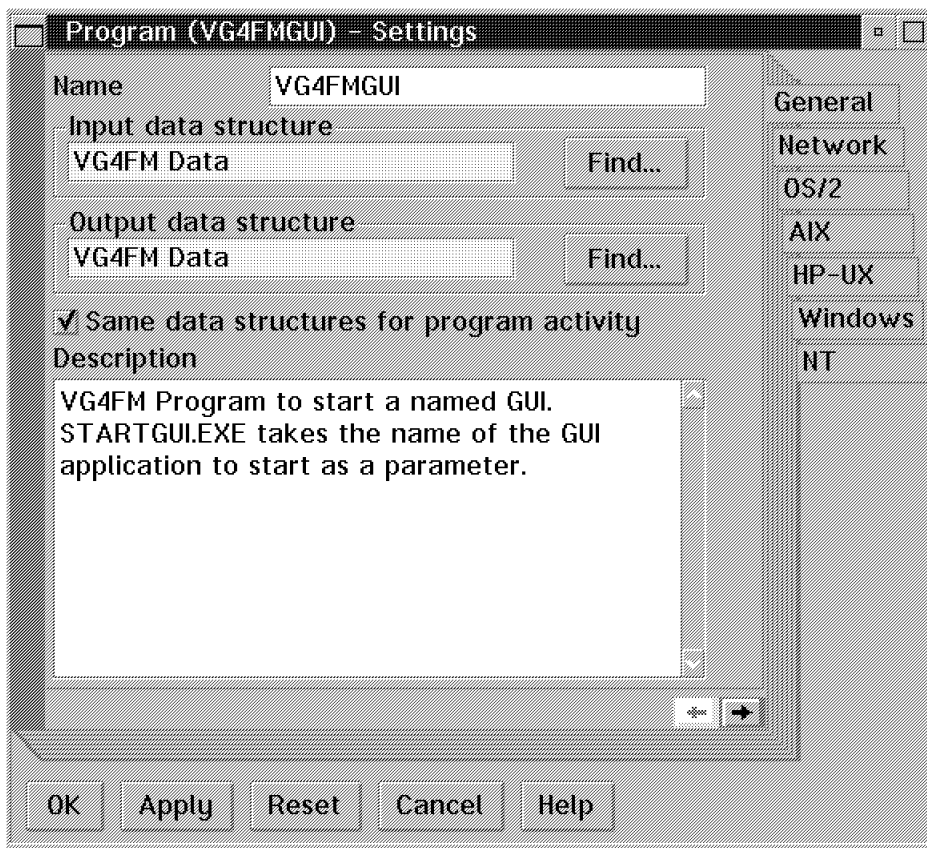


Figure 16. VG4FMGUI FlowMark Program Definition: General Settings

Figure 17 on page 25 shows the platform specific settings for Windows NT. The settings for OS/2 and Windows NT are identical for program VG4FMGUI.

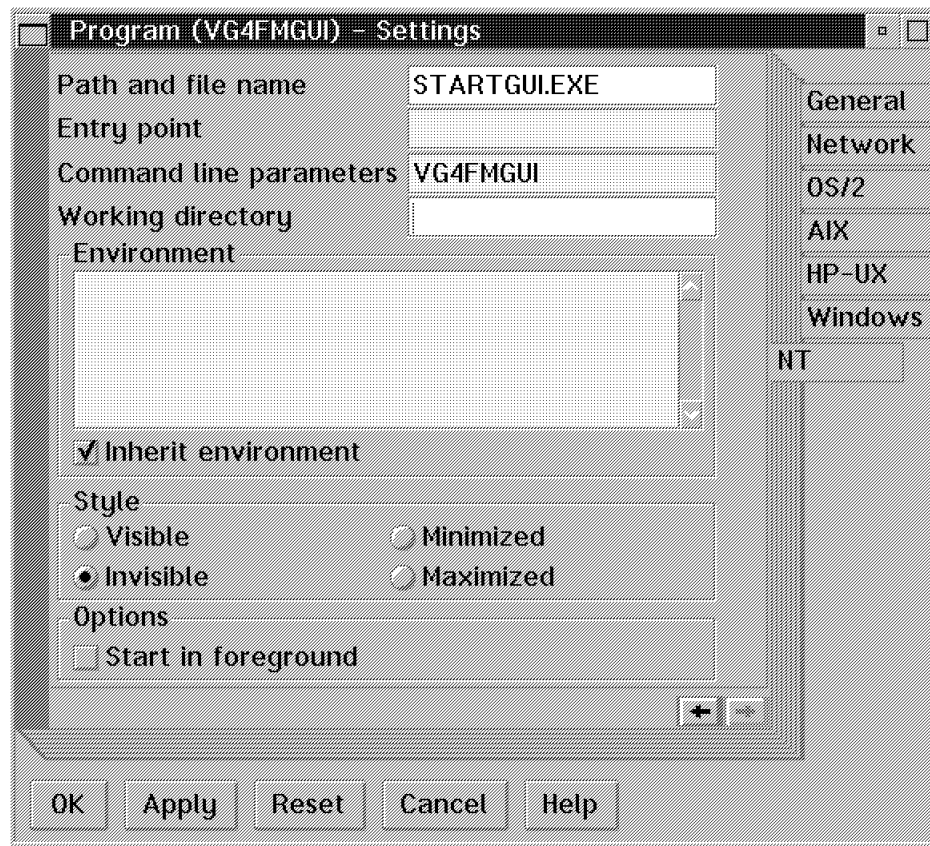


Figure 17. VG4FMGUI FlowMark Program Definition: NT Settings

After STARTGUI.EXE launches the GUI application by issuing the *EZE2RUN OPEN GUI application name* system command, it must wait until the GUI application terminates. If the GUI application terminates abnormally, it cannot notify STARTGUI.EXE that it has terminated. Therefore, STARTGUI.EXE must periodically check that the GUI application is running. The periodic check we have implemented in STARTGUI.EXE asks if the GUI application window is valid. If the window is not valid the GUI application has terminated.

In order to check the GUI window, STARTGUI.EXE must itself run in a window. In OS/2 this means that STARTGUI.EXE must be linked with the /PMTYPE:PM option. STARTGUI.EXE must have two threads, one for creating the STARTGUI.EXE window and handling the message queue and one for processing all communication with FlowMark and waiting for the GUI application to terminate. The code for the first thread is found in STARTGUI.C for OS/2 and STARTGUI.CPP for Windows NT. The code for the second thread is found in ST_GUI1.C and is the same for both OS/2 and Windows NT.

If the GUI application ends normally, it can notify STARTGUI.EXE that it has terminated and at the same time inform STARTGUI.EXE about the return code of the GUI application. You can choose to store the return code in the user-defined data structure member, RETURNCODE, through a FlowMark API call from either STARTGUI.EXE or VG4FMDLL.DLL. We implemented the FlowMark API call in VG4FMDLL.DLL because STARTGUI.EXE must be informed that the GUI application has terminated but does not need the return code of the GUI application.

If the GUI application terminates abnormally because of problems connecting to a host database, you may want to suspend the FlowMark workflow process instance until the connection to the host database is restored. We tried to implement the FlowMark API suspend call in STARTGUI.EXE, but it did not work.

If STARTGUI.EXE detects that the GUI application ended abnormally, it should exit with return code set to make it possible for the FlowMark workflow manager to use the predefined data structure member, _RC, for navigating the process path.

3.4 FlowMark-Enabled GUI Application

A VisualAge Generator GUI application that will be dispatched by and integrated with FlowMark must meet these criteria:

- It must be implemented as an external GUI application.
- It must incorporate the FlowMark-to-VisualAge Generator interface record and VG4FMDLL.DLL program calls.

Figure 18 shows the sample GUI application we created.

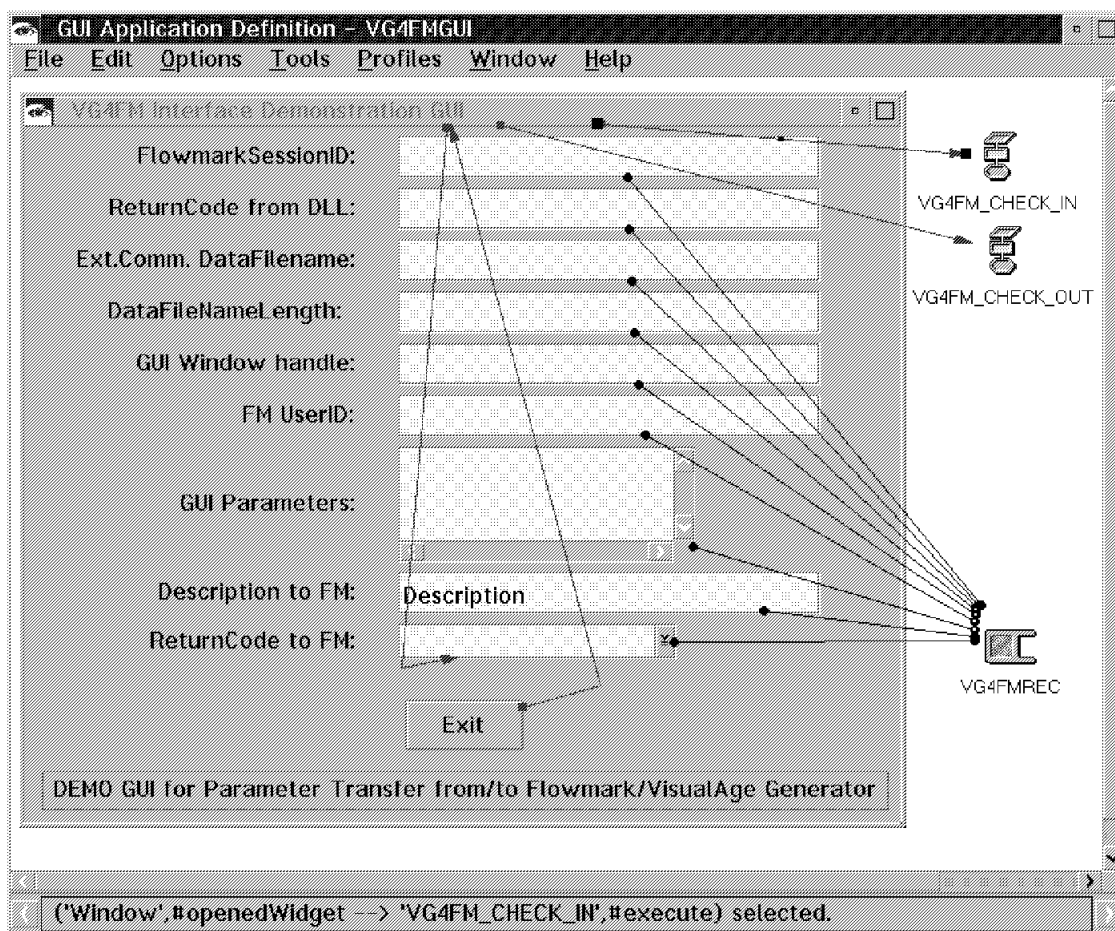


Figure 18. VisualAge Generator GUI Application Definition

The VisualAge Generator processes, VG4FM_CHECK_IN and VG4FM_CHECK_OUT, are triggered by the openedWidget and aboutToCloseWidget events, respectively.

When the GUI application calls CHECKIN or CHECKOUT, communication record VG4FMREC must be used as an argument. (Record VG4FMREC is described in 3.6, “FlowMark-VisualAge Generator Interface Communication Record” on page 29.)

Figure 19 on page 27 shows the VisualAge Generator process definition for VG4FM_CHECK_IN, which implements the call to VG4FMDLL.DLL function CHECKIN.

```

001 SET VG4FMREC EMPTY;      /* initialize VG4FMREC */
002
003 CALL CHECKIN VG4FMREC;

```

Figure 19. VisualAge Generator Process Definition for VG4FM_CHECK_IN

After calling CHECKIN, the GUI application can test the VG4FMREC RETURNCODE field to see whether the call was successful. If the call was successful, the values of the FM_USERID, PARAMETER, DATAFILENAME, and DATAFILENAMELENGTH fields are valid and can be used by the GUI application.

The GUI application can now do the task for which it was designed. When the task is done, the GUI application writes its return code in the RETURNCODE_TO_FLOWMARK field. The DESCRIPTION field can also be set. The GUI application now calls CHECKOUT, with VG4FMREC as argument.

Figure 20 shows the VisualAge Generator process definition for VG4FM_CHECK_OUT, which implements the call to the CHECKOUT function of VG4FMDLL.DLL.

```

001 /* VG4FMREC.RETURNCODE_TO_FLOWMARK must be set before calling CHECKOUT */
002 /* VG4FMREC.DESCRPTION may be set before calling CHECKOUT */
003
004 CALL CHECKOUT VG4FMREC;

```

Figure 20. VisualAge Generator Process Definition for VG4FM_CHECK_OUT

Because the GUI application used in a FlowMark workflow process calls the DLL CHECKIN and CHECKOUT functions, you must specify in a linkage table the names of the functions in the DLL, the name of the DLL (VG4FMDLL), and the link type for the GUI application call, so that the GUI application knows where to find the functions and how they must be called. A sample linkage table is provided on the diskette, in the VG4FM.LKG file, included with this redbook. You must use the linkage table as input when you generate the GUI application or use the Interactive Test Facility.

Figure 21 shows the linkage table for the CHECKIN and CHECKOUT functions.

```

:calllink applname=CHECKIN linktype=DYNAMIC library=VG4FMDLL
parmform=OSLINK .
:calllink applname=CHECKOUT linktype=DYNAMIC library=VG4FMDLL
parmform=OSLINK .

```

Figure 21. Linkage Table for CHECKIN and CHECKOUT Functions

3.5 FlowMark-VisualAge Generator Interface DLL

The FlowMark-VisualAge Generator interface DLL, VG4FMDLL.DLL, consists of two functions, CHECKIN and CHECKOUT. We include the implementation of both functions for both OS/2 and Windows NT in file VG4FMDLL.C on the diskette accompanying this redbook.

The GUI application must call CHECKIN when the GUI application window is active, that is, upon the openedWidget event, in order to retrieve the window handle. The window handle must be passed to STARTGUI.EXE, so that STARTGUI.EXE can detect whether the GUI application ends abnormally. Before the GUI application calls

CHECKIN, VG4FMREC is set to empty; after the call, CHECKIN has filled the fields in VG4FMREC. (Figure 19 shows the process definition for VG4FM_CHECK_IN.)

After CHECKIN has read the parameters supplied by STARTGUI.EXE in the C:\STRT_GUI.FIL file, it deletes the file. C:\STRT_GUI.FIL exists only during the launch of a GUI application—from when it is written by STARTGUI.EXE until it is deleted by VG4FMDLL.DLL. After C:\STRT_GUI.FIL has been deleted, another instance of STARTGUI.EXE can write its parameters to C:\STRT_GUI.FIL and launch a new GUI application. Thus, as long as C:\STRT_GUI.FIL exists, no instance of STARTGUI.EXE can launch a GUI application.³ In our implementation we chose to wait a maximum of 60 seconds before CHECKIN deleted C:\STRT_GUI.FIL, to ensure that if the system crashed during a GUI launch, the end user would not have to wait forever before a GUI was launched.

CHECKIN retrieves the GUI window handle through an operating-system-specific API call and passes it to STARTGUI.EXE. In our implementation, CHECKIN writes the GUI window handle to a file, which STARTGUI.EXE waits to read. If the name of the FlowMark workflow process instance is VG4FMGUI_61, CHECKIN writes the GUI window handle to the unique file C:\FMGUI_61.WIN.

Figure 22 shows the pseudo-code for the VG4FMDLL.DLL CHECKIN function.

```
Read GUI parameters in C:\STRT_GUI.FIL

Delete C:\STRT_GUI.FIL

Retrieve the GUI window handle

If OS/2, retrieve the GUI window anchor block handle

Save the handle(s) in a file or shared memory

Exit with return code to GUI application
```

Figure 22. Interface DLL CHECKIN Function Pseudo-Code Logic

After the GUI application has performed its task, it writes its return code in VG4FMREC and passes it to FlowMark through a call to CHECKOUT upon the aboutToCloseWidget event. If the GUI application wants the description changed, the DESCRIPTION field in VG4FMREC is also set to CHECKOUT before the call. (Figure 20 on page 27 shows how the GUI application calls CHECKOUT.)

CHECKOUT saves the GUI return code and description in the FlowMark user-defined data structure member, RETURNCODE, through API calls to FlowMark. If the GUI application terminates as a result of problems connecting to a host database, you might want to suspend the FlowMark workflow process instance until the connection to the host database is restored. Our implementation of CHECKOUT suspends the current FlowMark workflow process instance through a FlowMark API call if the RETURNCODE field in VG4FMREC is set to end abnormally (ABEND).

After the database connection has been restored, the end user can resume the FlowMark workflow process. You can provide the ability to restart a failed GUI application program activity if you put the GUI application program activity inside a block and use the user-defined data structure member, RETURNCODE, in the block's exit condition. The FlowMark workflow manager then automatically starts the GUI

³ This is important to remember when you implement our sample application. If you have problems getting the interface code to work, a copy of the C:\STRT_GUI.FIL file might get left on the drive. This file is created by INIT, used by STARTGUI, and erased by EXIT. After a failed start of the FlowMark-VisualAge Generator interface, the file must be manually erased before you can attempt to start the sample application again.

application when the FlowMark workflow process is resumed. You find this approach implemented in the FlowMark workflow process VG4FMGUI_Blocked (see Figure 12 on page 21).

CHECKOUT notifies the waiting instance of STARTGUI.EXE that the GUI application has terminated, by posting a semaphore, creating a shared memory object, or creating a file. In our implementation, we chose to use a file. If the name of the FlowMark workflow process instance is VG4FMGUI_61, CHECKOUT writes the GUI return code to the unique file C:\FMGUI_61.RC. When the file exists, STARTGUI.EXE resumes and can read the return code in the file.

Figure 23 shows the pseudo-code for the VG4FMDLL.DLL CHECKOUT function.

```

Save GUI return code in user-defined data structure member, RETURNCODE

If wanted, save GUI description in user-defined data structure member, DESCRIPTION

If wanted and if GUI return code indicates it, suspend process instance

Notify the waiting STARTGUI.EXE that the GUI application is terminated

Exit with return code to GUI application

```

Figure 23. Interface DLL CHECKOUT Function Pseudo-Code Logic

Because of the way in which the GUI application calls the DLL functions, you cannot use the default _Optlink calling convention when you generate the code for VG4FMDLL.DLL. If you use the IBM VisualAge C++ compiler, as we did for the samples, you can use the _System, __cdecl, or __stdcall calling convention, specifying option /Ms, /Mc, or /Mt. In the makefile for VG4FMDLL.DLL, VG4FMDLL.MAK, you can see that we used the _System (/Ms) calling convention.

3.6 FlowMark-VisualAge Generator Interface Communication Record

So far you know that the interface consists of an EXE program, STARTGUI.EXE, which communicates with FlowMark through API calls and the return code to the operating system, and a DLL, VG4FMDLL.DLL, which is called by the GUI application and communicates with FlowMark through API calls.

The GUI application communicates with the CHECKIN and CHECKOUT functions in VG4FMDLL.DLL through a communication record, *VG4FMREC*. In Table 1 we describe each field in communication record VG4FMREC.

Table 1 (Page 1 of 2). Description of the Fields in Record VG4FMREC	
Field Name	Description
FM_SESSIONID	Session identification of the FlowMark process instance in which the GUI is a program activity. Not used by and cannot be changed by the GUI application.
FM_USERID	The end user's FlowMark user ID that is passed to the GUI.
PARAMETER	Parameters that FlowMark passes to the GUI. The values of these parameters are set in the FlowMark process diagram (the name of the activity). The parameters can be used to enable or disable buttons or input fields.

Table 1 (Page 2 of 2). Description of the Fields in Record VG4FMREC	
Field Name	Description
DATAFILENAME	Name of the file that a GUI in the current FlowMark process instance can use for reading data from a previous GUI or for writing data to be used in a subsequent GUI.
DATAFILELENGTH	Length of the DATAFILENAME
RETURNCODE	Return code from the CHECKIN or CHECKOUT function. You can define your own values. A value of 0 usually indicates that a call terminated successfully.
RETURNCODE_TO_FLOWMARK	The return code of the GUI to be passed to FlowMark, indicating in which condition the GUI terminated. Used by FlowMark workflow manager to navigate the process path.
DESCRIPTION	A description of this FlowMark process instance, which can be shown in the FlowMark runtime Work List folder. The description can help the user to distinguish instances of the same process.
WINDOW_HAB	The anchor block handle of the GUI window. Not used by the GUI application.
WINDOW_HANDLE	The handle of the GUI window. Not used by the GUI application.
RESERVED	Reserved for future use. Also serves to protect the return address on the stack when the GUI calls CHECKIN or CHECKOUT. Not used by the GUI application.

VG4FMREC must be defined in the same way in both VG4FMDLL.DLL and the GUI application. The fields in the record must match exactly; otherwise the contents of the fields will be interpreted incorrectly by the GUI application or CHECKIN and CHECKOUT. As the record is aligned on a byte boundary in a GUI application, you must ensure that the C definition of the record is byte aligned as well by specifying the *#pragma pack(1)* directive.

Figure 24 on page 31 shows the definition of the FlowMark-VisualAge Generator communication record as it is defined in the VG4FMDLL.DLL. The C language definition is included in the VG4FMINC.H file on the diskette accompanying this redbook.

```

#pragma pack(1)                                /* align on byte boundary */

typedef struct
{
    char pszFM_SessionID          [ 20 ];
    char achFM_UserId             [  8 ];
    char achParmeter              [3][ 20 ];
    char achDataFileName          [ 65 ];
    char achDataFileNameLenght    [  2 ];
    char achReturnCode            [  3 ];
    char achReturnCodeToFlowMark  [ 30 ];
    char achInstanceDescription   [ 20 ];
#ifdef __TOS_OS2__                /* target operating system is OS/2 */
    HAB habGUI;                    /* 4 */
#else
    HWND hwndGUI2;
#endif /* __TOS_OS2__ */
    HWND hwndGUI;                  /* 4 */
    char achFutureUse             [ 25 ];
} VG4FMREC;

#pragma pack( )                                /* restore to system default alignment */

```

Figure 24. Definition of Communication Record VG4FMREC in C Language

Figure 25 shows the definition of the FlowMark-VisualAge Generator communication record as it is defined in the VisualAge Generator Developer. The definition, in external source format, is included in the VG4FMGUI.ESF file on the diskette accompanying this redbook.

NAME	LVL	OCCURS	TYPE	SCOPE	LENGTH	DEC	BYTES	START	MSL	DESCRIPTION
FM_SESSIONID	10		CHAR	LOCAL	20		20	1		Flowmark session ID
FM_USERID	10		CHAR	GLOBAL	8		8	21	VG4FM	FlowMark userid
PARAMETER	10	3	CHAR	LOCAL	20		20	29		Parameters for controlling GUI
DATAFILENAME	10		CHAR	LOCAL	65		65	89		DateFilename for external comm
DATAFILENAMELENGTH	10		NUM	LOCAL	2		2	154		Length of full datafilename
RETURNCODE	10		NUM	LOCAL	3		3	156		Returncode from CHECKIN/OUT
RETURNCODE_TO_FLOWMARK	10		CHAR	LOCAL	30		30	159		GUI's returncode to FlowMark
DESCRIPTION	10		CHAR	LOCAL	20		20	189		GUI's description of instance
WINDOW_HAB	10		HEX	LOCAL	8		4	209		Anchorblockhandle of GUIwindow
WINDOW_HANDLE	10		HEX	LOCAL	8		4	213		Handle of GUI window
RESERVED	10		CHAR	LOCAL	25		25	217		For future use

Figure 25. Definition of Communication Record VG4FMREC in VisualAge Generator Developer

The length of each field must be the same in both definitions. The C language definition is valid in both OS/2 and Windows NT. The only difference is that under OS/2 you must have an anchor block handle when retrieving the GUI window handle. Therefore, hwndGUI2 is declared as a dummy placeholder under Windows NT, so that the record has the same length whether it is compiled under OS/2 or Windows NT.

Notice that the scope of FM_USERID is global, so it is possible to use FM_USERID throughout the entire GUI application.

Chapter 4. VisualAge Generator and FlowMark Sample Application

A sample application that demonstrates the use of the code discussed in Chapter 3, "Implementing a FlowMark-to-VisualAge Generator Interface" on page 19 is included on the diskette shipped with this redbook. The sample application consists of:

- C/C++ interface program materials
- VisualAge Generator GUI application source and a generated .APP file
- FlowMark process definitions in .FDL and .FRL files

The sample application demonstrates how VisualAge Generator GUI applications can be integrated into a FlowMark-based system. The sample application can be implemented and run on either an OS/2 or a Windows NT platform.

In this chapter we provide information on the installation and configuration of the prerequisite software and the use of the sample application. Platform-specific requirements for OS/2 and Windows NT are discussed in each section.

Note: This chapter is not a substitute for appropriate VisualAge Generator and FlowMark configuration and administration skills. We only outline the steps required.

4.1 Prerequisite Software

The software prerequisites and configuration process for OS/2 and Windows NT are very similar. A full runtime system can be established on one workstation using either OS/2 or Windows NT.

Several products must be installed on your selected workstation platform before you begin to use the sample application code we provide on the diskette.

- On an OS/2 workstation, you need:
 - OS/2 Warp Version 3
 - VisualAge Generator GUI Runtime Support Version 2.2 (FixPak 3)
 - IBM VisualAge C++ for OS/2 Version 3.0
 - IBM FlowMark Version 2.3
- On a Windows NT workstation, you need:
 - Windows NT Version 3.51 or 4.00
 - VisualAge Generator GUI Runtime Support Version 2.2 (FixPak 3)
 - IBM VisualAge C++ for Windows Version 3.5⁴
 - IBM FlowMark Version 2.3 (with FixPak 1 if you want to compile the sample application C/C++ code)

⁴ Our code may work (with some modifications) with other C++ compilers, we did not test this.

The implementation of the runtime system on a Windows NT platform may require the use of an OS/2 system if you need to perform these optional tasks:

- Generate VisualAge Generator GUI application

We have provided a copy of the generated GUI application .APP file on the diskette, but it will work for you only if you have VisualAge Generator GUI Runtime Support V2.2 installed at the FixPak 3 level.⁵

If you need to generate the GUI application, you must have access to VisualAge Generator Developer V2.2. Version 2.2 of VisualAge Generator Developer is available on the OS/2 platform only.

- Import FlowMark buildtime definitions and translate to create FlowMark runtime definitions

We have provided a FlowMark runtime language definition file (.FRL) that can be imported into the FlowMark ObjectStore database to implement the runtime environment. This file does not provide access to the FlowMark buildtime definitions, which represent the development image of a FlowMark process. However, buildtime definitions can be created if you use the FlowMark runtime client to import the provided FlowMark definition language (.FDL) file.

Version 2.3 of FlowMark provides support for implementing a runtime client (and .FDL file import) on the OS/2 platform only. The OS/2 FlowMark runtime client can communicate with the Windows NT FlowMark ObjectStore database.

Buildtime definitions can be translated into runtime definitions using the FlowMark runtime client.

The sample application system can be implemented on a Windows NT platform without performing these optional tasks if you have the right software installed (VisualAge Generator GUI Runtime Support V2.2 with FixPak 3).

4.2 Implementation

There are several mandatory and optional sample application implementation tasks:

1. Unzip sample application diskette files.

Install the sample application files provided in a ZIP file on the diskette on your hard disk drive by running the command specified in the readme file on the diskette.

This will create the directory VG4FM on your target drive.

The VG4FM directory contains these subdirectories:

INIT	Contains the INIT program source materials
EXIT	Contains the EXIT program source materials
STARTGUI	Contains two subdirectories (WINNT and OS2) for each target platform
VG4FMDLL	Contains two subdirectories (WINNT and OS2) with VG4FMDLL program source materials for each target platform
INCLUDE	Contains program source materials used in the VG4FMDLL program
ESF	Contains a VisualAge Generator GUI application definition in external source format and the linkage table required for generation
FLOWMARK	Contains buildtime (.FDL) and runtime (.FRL) definition files

⁵ The .APP file may work with VisualAge Generator GUI Runtime Support V2.2 at other FixPak levels, but we did not test this scenario.

DLL	Contains two subdirectories (WINNT and OS2) with a compiled version of the VG4FMDLL program
EXE	Contains two subdirectories (WINNT and OS2) with compiled versions of the INIT, STARTGUI, and EXIT programs
VGGUI	Contains a generated version of the VisualAge Generator GUI application

These sample application files will be used during the remaining implementation steps.

2. Install VisualAge C++.

Steps for OS/2:

- We selected all available components of C++ during the installation process. Not all of the components are required, but we did not experiment to determine which we could leave out.
- We chose a target drive and directory of E:\IBMCP.
- We rebooted the operating system to complete the workframe installation process.

Steps for Windows NT:

- We selected the *typical* C++ installation option during the the installation process. Not all of the components are required, but we did not experiment to determine which we could leave out.
- We chose a target drive and directory of D:\IBMCPW and said yes to the question about creating the directory.
- We did not reboot the operating system until after the FlowMark installation task (Install FlowMark).

3. Install VisualAge Generator GUI Runtime Support V2.2 with FixPak 3

Steps for OS/2:

- To install the base level of VisualAge Generator GUI Runtime Support V2.2, we started INSTALL.EXE in the \EZERDEV2\ENU subdirectory of the VisualAge Generator V2.2 CD. This is the install program for OS/2.
- We clicked on the **Continue** push button, and then on the next dialog window selected **Update CONFIG.SYS** and clicked on the **OK** push button.
- We selected just the VisualAge Generator GUI Runtime Support feature and chose target drive and directory values of D:\EZERDEV2 and D:\EZERDEV2\TEMP.
- We then installed FixPak 3 by starting INSTALL.EXE in the \EZERWIN\ENU subdirectory of the FixPak directory tree.
- We told the installation program to update the currently installed products (which for us on this workstation only included VisualAge Generator GUI Runtime Support V2.2), to update CONFIG.SYS, and not to save a backup version of the installed product.
- We did not reboot the operating system until we completed step 4 (Install FlowMark).

Steps for Windows NT:

- To install the base level of VisualAge Generator GUI Runtime Support V2.2, we started SETUP.EXE in the \EZERWIN\ENU subdirectory of the VisualAge Generator V2.2 CD. This is the install program for Windows 95 and Windows NT.
- We chose target drive and directory values of D:\EZERUN and D:\EZEWIN.

- We did not reboot the operating system at the end of the installation process.
- We then installed FixPak 3 by starting SETUP.EXE in the \EZERWIN\ENU subdirectory of the FixPak directory tree.
- The installation program remembered our target drive and directory values of D:\EZERUN and D:\EZEWIN.
- We did not reboot the operating system until we completed step 4 (Install FlowMark).

4. Install FlowMark.

Steps for OS/2:

- We selected all available FlowMark components, except for the **Standalone Installation** option. (We wanted to be able to connect to the machine later.)
- We chose a target drive and directory of E:\EXM.
- FlowMark did not find a previous version on our system, and we did not have FlowMark ObjectStore databases we wanted to migrate, so we said *continue* at the prompt.
- We provided the following configuration information:

Database Client Settings:

Local Host:	ireland (our TCP/IP host name)
Database Directory:	E:\EXM
Database Name:	EXMDB
UNIX User ID/Comp ID:	Did not change (was 9 and 999)
ObjectStore Network Protocol:	Local was grayed out and selected; we selected TCP/IP.

FlowMark Server Settings:

Server Name:	EXMSRV
Network Protocols:	TCP/IP and Local

FlowMark Client Settings:

FlowMark Server Name:	EXMSRV
Database Name:	EXMDB
Protocol Type:	TCP/IP with an IP address of 129.33.160.59

We accepted the default VisualAge settings, even though we did not use VisualAge Smalltalk.

- After rebooting the operating system, we had to initialize the FlowMark ObjectStore database. But first we had to rename a .DLL file installed as part of C++ (see Figure 26 on page 37).

From the FlowMark README.ENU file:

2. Coexistence of ObjectStore 4.x and VisualAge C++ Version 3

If IBM VisualAge C++ for OS/2 V3 is installed, the ossserver executable (FlowMark Database Server) occasionally reports a general protection fault.

Ossserver -i (Database Server initialization) is invoked as part of the FlowMark installation procedures.

To solve the problem, remove the VisualAge entry \IBMCPPL\DLL from the Config.Sys LIBPATH statement or rename the DDE4MTH.DLL.

Figure 26. FlowMark and C++ Coexistence Issue

- We renamed the DDE4MTH.DLL file and then ran the command to initialize the ObjectStore database (see Figure 27).

```
[C:\]jossserver -i
970409 111641 ObjectStore Release 4.0.2 Database Server
970409 111642 LOG 0001 There are no partitions specified in the parameters file.

Only file databases will be accessible through this server.
970409 111642 The transaction log file is E:\EXM\LOG\EXM.LOG

You have asked for initialization which will
create a new transaction log, deleting any
old log that may exist, thus destroying any
recovery data in the old log. This may leave
some file databases in the broken state. Are you
sure that you want to create a new log? (yes/no): yes

[C:\]
```

Figure 27. FlowMark ObjectStore Database Initialization for OS/2

Steps for Windows NT:

- We selected all available FlowMark components for installation.
- We chose a target drive and directory of D:\EXMWINNT and said yes to the request to create the directory.
- We provided the following configuration information:

Installation Settings:

Local Computer Name:	bismuth (our TCP/IP host name)
Database Name:	EXMDB
FlowMark Server Name:	EXMSRV
Protocol:	Local was grayed out and selected; we selected TCP/IP.

Language Group:

We chose Latin 1.

Select Program Folder:

We chose the name FlowMark V2.3.

- After copying the files, the FlowMark install process started the ObjectStore database initialization task. We pressed a key in response to the initialization text to complete the task.

Additional information about FlowMark installation, configuration, and software validation is available in *IBM FlowMark: Installation and Maintenance, Version 2 Release 3*.

5. Build sample application C/C++ executables (optional).

Note:

Compiled versions of our C/C++ executables for OS/2 and Windows NT are available in the \EXE and \DLL subdirectories. You do not have to build the C/C++ programs if you have installed the software defined in 4.1, "Prerequisite Software" on page 33. So, if you want, just skip to the next step.

- The `OS2_SHELL` environment variable, set in the CONFIG.SYS file of an OS/2 system, is used by some of our C/C++ makefiles to control the compilation process. This enabled us to use the same makefiles for OS/2 and Windows NT for some of the C/C++ programs. Check that the `OS2_SHELL` environment variable has a value on your OS/2 system.
- Certain environment variables must contain references to FlowMark product directories before we can build the C/C++ executables. These are set in CONFIG.SYS during the FlowMark installation for OS/2, but we had to add them on Windows NT:

INCLUDE On OS/2, E:\EXM\API and E:\EXM\SBM\INCLUDE
 On Windows NT, D:\EXMWINNT\API

LIB On OS/2, E:\EXM\API and E:\EXM\SBM\LIB
 On Windows NT, D:\EXMWINNT\API

- We built the C/C++ executables by changing to the following program directories and issuing the NMAKE command:

For OS/2: \VG4FM\INIT
 \VG4FM\STARTGUI\OS2
 \VG4FM\VG4FMDLL\OS2
 \VG4FM\EXIT

For Windows NT: \VG4FM\INIT
 \VG4FM\STARTGUI\WINNT
 \VG4FM\VG4FMDLL\WINNT
 \VG4FM\EXIT

Note:

To build the executables on Windows NT, we had to first install FlowMark FixPak 1. The Windows NT executables we provide were compiled on a system with FixPak 1, but they will run on a GA level FlowMark V2.3 installation.

6. Make the C/C++ executables available in system directories.

Note:

Either the versions that were built in step 5 or the versions provided in the sample application \EXE and \DLL subdirectories can be copied.

- We copied the INIT.EXE, EXIT.EXE, and STARTGUI.EXE executable programs to a directory in the PATH environment variable.
 - We copied the VG4FMDLL.DLL program to a directory in the LIBPATH environment variable (OS/2) or the PATH environment variable (Windows NT).
 - We copied the help file for the STARTGUI program (STARTGUI.HLP) to a directory in the HELP environment variable (Windows NT only).
7. Generate VisualAge Generator GUI application (optional).
 - We imported the VisualAge Generator GUI application source provided in the \ESF\VG4FMGUI.ESF file to an MSL.
We used the linkage table provided when we generated the GUI application (/LINKAGE=d:\VG4FM\ESF\VG4FMGUI.LKG).
 8. Make the VisualAge Generator GUI application available in system directories.
 - We copied the generated GUI application (.APP file) to a directory in the DPATH environment variable.
 9. Start FlowMark.
 - We used the *Startup FlowMark* icon in the FlowMark folder to start the FlowMark environment.
We started these servers:
 - Database server (OS/2 only)⁶
 - Runtime server
 - Program execution client
 We did not need to start these servers:
 - Bundle service
 - Notification service
 - Delivery server
 - The first time FlowMark is started, you get prompts asking you if you want to create the EXMDB database (we said *yes*) and if you want to create the EXMSRV server in the EXMDB database (we said *yes*).
 10. Import FlowMark buildtime definitions and translate them to create runtime definitions (optional).

Note:

You can perform task 10 and skip task 11 (Import FlowMark runtime definitions) or skip task 10 and perform task 11. To run the sample application system, you must perform either task 10 or 11.

Steps for OS/2:

- We started the FlowMark buildtime process (Buildtime icon in the FlowMark folder).
- We logged on to the FlowMark environment using user ID and password values of admin and password. The EXMDB database should already be identified in the *FlowMark Logon Buildtime* dialog.

⁶ On Windows NT this option cannot be selected. On Windows NT the ObjectStore database server is registered as a service that is started by the operating system.

- We started the *Buildtime Processes* icon in the Buildtime folder. Even if you have imported the FlowMark runtime definitions, this folder will be empty. Buildtime and runtime definitions are stored in different areas in the FlowMark ObjectStore database.
- We closed the Buildtime Processes view and started the *Import* icon in the Buildtime folder.
- We identified the file we wanted to import (D:\VG4FM\FLOWMARK\VG4FM2.FDL), selected the **Write into current database** toggle button and the **Ask for a decision** radio button, and then clicked on the **OK** push button to start the import.
- We clicked on **Yes** to start the import.
- We started the *Buildtime Processes* icon in the Buildtime folder. Two processes should now be showing. Program and data structure definitions can also be found if you start (open) the Programs and the Data Structures views.

These are the development versions of the sample application FlowMark definitions. To use them in a runtime environment, we must translate the process definitions.
- We selected both processes in the Buildtime Processes view and chose the **Translate / New template** option. This can be done by using either the context menu (click mouse button 2) or the **Selected** option on the action bar. If you have previously translated the definitions, you may want to replace the templates. The **New templates** option will create additional templates with modified process names.

We are now ready to run the sample application. Instructions are provided in 4.3, "Runtime Operation" on page 42.

Steps for Windows NT:

Note:

If you are implementing the sample application on a Windows NT platform and want to perform task 10, you have to first set up an OS/2 FlowMark buildtime client workstation.

- Set up an OS/2 buildtime Client workstation:
 - During installation we selected the *Buildtime Client* component.
 - We chose a target drive and directory of E:\EXM.
 - FlowMark did not find a previous version on our system and we did not have FlowMark ObjectStore databases we wanted to migrate, so we said *continue* at the prompt.
 - We provided the following configuration information:

Database Client Settings:

Local Hostname:	cork (our TCP/IP host name)
Database Server Hostname:	bismuth (the TCP/IP host name for the Windows NT workstation)
Database Directory:	d:\EXMWINNT (The target is the Windows NT workstation.)
Database Name:	EXMDB
Unix User ID/Comp ID:	Did not change (was 9 and 999)
ObjectStore Network Protocol:	Named Pipes was grayed out and selected; we also selected TCP/IP.

FlowMark Server Settings:

Server Name: EXMSRV (predefined and grayed out)
Network Protocols: TCP/IP and Local

- We rebooted the operating system to complete the installation.
- We also checked the FlowMark preferences file (E:\exm\bin\exmpzcfg.prf) to ensure that the settings reflected the answers provided during installation.
- We checked that we had a TCP/IP connection to the target FlowMark server workstation, using the ping bismuth command.
- We imported the FlowMark buildtime definitions, using the instructions provided above for OS/2. The FlowMark client stores the information in the Windows NT ObjectStore database.

Note:

If you log on from an OS/2 buildtime client to a Windows NT FlowMark server you will get a message about code-page differences between the client and server. For our purposes, and to run the sample application, this was not an issue.

11. Import FlowMark Runtime definitions.

- We used the `exmpfrim` command to import the sample application FlowMark runtime definitions provided in the D:\VG4FM\FLOWMARK\VG4FMDB.FRL file (see Figure 28 on page 42).

```

[E:\vg4fm\flowmark]expfrim
Specify file name for import (default is "EXMPF$$$"): vg4fmdb.fr1
Initializing...

The ObjectStore server on host ireland is alive.
Database (default is "EXMDB"):
UserID   (default is "ADMIN"):
Password : *****

Phase 1: Parsing input file .

    0 .....
Completing definitions.
Importing temporary buildtime objects.
.....
Translation of PROCESS VG4FMGUI was successful.
Restoring original names.
.....
Deleting temporary buildtime objects.
.....
.....
Completing definitions.
Importing temporary buildtime objects.
.....
Translation of PROCESS VG4FMGUI-Blocked was successful.
Restoring original names.
.....
Deleting temporary buildtime objects.
.....

Import/Export found 0 error(s), 0 warning(s).

[E:\vg4fm\flowmark]

```

Figure 28. Importing FlowMark Runtime Definitions. The process is identical for OS/2 and Windows NT. The default FlowMark user ID and password values are ADMIN and PASSWORD.

We are now ready to run the sample application.

4.3 Runtime Operation

Once the required setup tasks have been completed (see 4.2, "Implementation" on page 34), we can log on to the FlowMark runtime client and run the sample application.

We followed these steps as a guide through the process of using both the VG4FMPCRC or VG4FMPCRC-Blocked processes:

1. We started the FlowMark runtime client.
2. We opened the processes folder.
3. We then ran the VG4FMPCRC and VG4FMPCRC-Blocked processes:
 - a. We first created an instance of the process (you can double-click on the process template to do this).
 - b. We then started the instance of the process (you can double-click on the process just created from the template to do this).

If the FlowMark or VisualAge Generator GUI runtime environment has not been set up properly, this task could fail. Review the log files created by the FlowMark-to-VisualAge Generator interface code to help identify the problem. We had some initial trouble getting the call to VG4FMDLL.DLL from the GUI application to work at first. We had to erase the C:\STRT_GUI.FIL file before trying to start the process again (see 3.5, “FlowMark-VisualAge Generator Interface DLL” on page 27).

- c. To end the GUI application we provided a return code value in the VisualAge Generator GUI application and clicked on the **Exit** push button.
We performed these steps twice, once with an *ABEND* return code and once with an *OK* return code.
- d. When we inspected the view of the process instance in the FlowMark process folder, we noticed it was still there.
- e. We restarted (resumed) the process using the process context menu. Only the VG4FMPC-Blocked process will restart the GUI application.
- f. This time we provided an *OK* return code value in the VisualAge Generator GUI application, and clicked on the **Exit** push button.

Figure 29 shows the VisualAge Generator GUI application started by the FlowMark-to-VisualAge Generator interface program.

FlowmarkSessionID:	E29
ReturnCode from DLL:	0
Ext.Comm. DataFilename:	C:\locked_2.DAT
DataFileNameLength:	15
GUI Window handle:	%0
FM UserID:	ADMIN
GUI Parameters:	- VG4FMGUI - PARM11 - PARM12
Description to FM:	
ReturnCode to FM:	ABEND

Exit

DEMO GUI for Parameter Transfer from/to Flowmark/VisualAge Generator

Figure 29. Runtime View of VG4FMGUI Application

Appendix A. GUI Application Terms and Implications

In this publication we use several terms to describe GUI applications. A single GUI application could be described as primary, functional, and external at the same time. Although a primary GUI application is a functional GUI application, not all functional GUI applications are primary GUI applications, but both functional and primary GUI applications are external GUI applications. To help you interpret our terms as you read this document we define them here:

Embedded GUI application

A GUI application whose primary part is not a window. It cannot be opened for display independently of another GUI application. To use an embedded GUI application, you have to include it in another GUI application.

External GUI application

A GUI application whose primary part is a window. It can be opened for display independently of another GUI application. External GUI applications can, and often do, add other embedded or external GUI applications to their free-form surface.

Functional GUI application

An external GUI application that can be started and implement a given set of processing as required by the application system. A GUI application that implements support for list or detail view processing on an object is a functional GUI application. An external GUI application that provides message processing support is not a functional GUI application because it does not provide direct functional support to the end user for a specific object.

Primary GUI application

A functional GUI application that can be directly invoked from the entry point. A functional GUI application that can be invoked only from another functional GUI application is not a primary GUI application.

The role of functional and primary GUI applications and the different types of GUI applications (entry point, list, detail) are discussed in *Object-Based GUI Application Development with VisualGen*, SG24-4233, the source of these definitions.

Appendix B. Special Notices

This publication is intended to help system architects and developers develop solutions to business problems by using the combined capabilities of VisualAge Generator and FlowMark. The information in this publication is not intended as the specification of any programming interfaces that are provided by VisualAge Generator or FlowMark. See the PUBLICATIONS section of the IBM Programming Announcement for VisualAge Generator and FlowMark for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 51.

- *Image and Workflow Library: FlowMark V2.2 Design Guidelines*, SG24-4613-00

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *IBM FlowMark: Programming Guide*, SH19-8240-02
- *IBM FlowMark: Installation and Maintenance, Version 2 Release 3*, SH12-6260-01
- *Developing VisualAge Generator Client/Server Applications*, SH23-0230-00

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL <http://www.redbooks.ibm.com>.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

To get LIST3820s of redbooks, type one of the following commands:

```
TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
```

To get BookManager BOOKs of redbooks, type the following command:

```
TOOLCAT REDBOOKS
```

To get lists of redbooks, type one of the following commands:

```
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
```

To register for information on workshops, residencies, and redbooks, type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
```

For a list of product area specialists in the ITSO: type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
```

- **Redbooks Home Page on the World Wide Web**
<http://w3.itso.ibm.com/redbooks>
- **IBM Direct Publications Catalog on the World Wide Web**
<http://www.elink.ibm.link.ibm.com/pb1/pb1>

IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibm.link.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	Internet
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark
--	--	--

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States) or (+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Home Page <http://www.redbooks.ibm.com>
"Redpieces" - online-readable excerpts from redbooks in progress
<http://www.redbooks.ibm.com/redpieces>
IBM Direct Publications Catalog <http://www.elink.ibm.link.ibm.com/pbl/pbl>

- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibm.link.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name
------------	-----------

Company

Address

City	Postal code	Country
------	-------------	---------

Telephone number	Telefax number	VAT number
------------------	----------------	------------

- Invoice to customer number

- Credit card number

Credit card expiration date	Card issued to	Signature
-----------------------------	----------------	-----------

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Special Characters

- _ACTIVITY**
 - predefined data structure member 19
 - user-defined data structure member 19
- _RC**
 - predefined data structure member 19
 - user-defined data structure member 19

A

- aboutToCloseWidget
 - DLL interface 14

B

- batch applications 2
- blocks 3
- buildtime process
 - diagramming facility 3

C

- C:\STRT_GUI.FIL 22, 28
- CHECKIN 26, 27
- CHECKOUT 26, 27
- communication
 - C:\FMGUI_61.DAT 23
 - C:\STRT_GUI.FIL 22
 - file-based 15, 22, 23
 - implementation 22, 23
 - overview 15
- containers 3
- control connectors 3

D

- data connectors 3
- data structures 3
- DDE interface
 - communication 17
 - GUI application 17
 - overview 15
 - STARTDDE.EXE 16

DDE4MTH.DLL 37

DLL interface

- aboutToCloseWidget 14
- C:\FMGUI_61.DAT 23
- C:\STRT_GUI.FIL 22, 28
- CHECKIN 14, 26, 27
- CHECKOUT 14, 26, 27
- communication 15
- file-based 15
- FlowMark workflow manager definitions 19
- GUI application 14, 26
- implementation 19
- INIT.EXE 20, 22
- openedWidget 14
- overview 13
- program activity
 - INIT 20
 - INIT.EXE 20, 22
 - STARTGUI.EXE 20, 22, 23
 - VG4FMGUI 20, 23
 - STARTGUI.EXE 14, 22
- synchronization 22
- VG4FMDLL.DLL 14
- VG4FMREC 29

E

- exit conditions
 - example 4
 - introduction 3
 - runtime resolution 3
- EXIT.EXE
 - program activity 20
 - sample application 34
- EZE2RUN 8

F

- FlowMark
 - blocks 3
 - buildtime process 3
 - containers 3
 - control 3
 - data connectors 3

FlowMark (*continued*)

- data structures 3
- DDE interface 15
- diagramming facility 3
- DLL interface 13
- exit conditions 3
- interface 6
- interface implementation 19
- interface issues 7
- interface objectives 11
- interface options 12
- introduction 3
- launching GUI applications 8
- logon 6
- predefined data structure member 4
- processes 3
- program activity 3, 11
- programs 3
- servers 3
- staff 3
- support for GUI applications 8
- support for programs 7
- transition conditions 3
- workflow 3
- workflow manager 6

FlowMark workflow manager

- _ACTIVITY 19
- _RC 19
- definitions 19
- predefined data structure member 19
- process diagram 20
- user-defined data structure member 19

G

GUI application

- aboutToCloseWidget 14
- CHECKIN 14, 26
- CHECKOUT 14, 26
- command file 8
- DDE interface 17
- description 45
- DLL interface 14
- embedded 45
- embedded GUI application 2
- entry point 45
- external 45
- external GUI application 2
- functional 2, 45
- implementation 26
- introduction 2
- launching 8
- linkage table 27
- openedWidget 14
- primary 2, 45
- terms 45
- types 2, 45
- VG4FM_CHECK_IN 26

GUI application (*continued*)

- VG4FM_CHECK_OUT 26
- VG4FMREC 29

I

INIT.EXE

- implementation 22
- program activity 20
- sample application 34

integration

- DDE interface 15
- DLL interface 13
- GUI application 26
- implementation 19
- interface issues 7
- interface objectives 11
- interface options 12

interface

- DDE 15
- DLL 13
- FlowMark workflow manager definitions 19
- GUI application 26
- implementation 19
- integration issues 7
- integration objectives 11
- integration options 12
- introduction 1, 6
- objectives 1

L

linkage table 27

logon

- FlowMark 6

O

ObjectStore

- C++ coexistence 37
- DDE4MTH.DLL 37
- initialization 36

openedWidget

- DLL interface 14

P

predefined data structure member

- _ACTIVITY 19
- _RC 4, 19
- DLL interface 19

processes

- description 3

program

- settings 5, 6

- program activity
 - EXIT 20
 - EXIT.EXE 20
 - INIT 20
 - INIT.EXE 20, 22
 - introduction 3
 - settings 5
 - STARTGUI.EXE 22
 - states 11
- project
 - introduction 1
 - objectives 1

R

- return code
 - user-defined data structure member 25, 27

S

- sample application
 - EXIT.EXE 34
 - implementation 34
 - INIT.EXE 34
 - prerequisite software 33
 - runtime operation 42
 - setup 33
 - STARTGUI.EXE 34
 - VG4FMDLL.DLL 34
- servers 2, 3
- staff 3
- STARTDDE.EXE
 - overview
 - DDE interface 16
- STARTGUI.EXE
 - C:\FMGUI_61.DAT 23
 - C:\STRT_GUI.FIL 22
 - implementation 22
 - overview
 - DLL interface 14
 - program activity 20, 23
 - sample application 34
 - synchronization 22
 - VG4FMGUI 20, 23
- synchronization
 - C:\STRT_GUI.FIL 22, 28
 - communication 22
 - DLL interface 22
 - file-based 22

T

- text user interfaces 2
- transition conditions
 - example 4
 - runtime resolution 3

U

- user-defined data structure member
 - _ACTIVITY 19
 - _RC 19
 - description 19
 - DLL interface 19
 - return code 25, 27

V

- VG4FM_CHECK_IN 26
- VG4FM_CHECK_OUT 26
- VG4FMDLL.DLL
 - CHECKIN 14, 26, 27
 - CHECKOUT 14, 26, 27
 - implementation 27
 - overview 14
 - sample application 34
 - VG4FMREC 29
- VG4FMREC 29
- VisualAge Generator
 - batch applications 2
 - DDE interface 15
 - DLL interface 13
 - GUI applications 2
 - interface 6
 - interface implementation 19
 - interface issues 7
 - interface objectives 11
 - interface options 12
 - introduction 2
 - launching GUI applications 8
 - servers 2
 - text user interfaces 2

W

- workflow
 - blocks 3
 - buildtime process 3
 - containers 3
 - control 3
 - data connectors 3
 - diagramming facility 3
 - elements included 3
 - exit conditions 3
 - program activity 3
 - transition conditions 3
- workflow management 1

ITSO Redbook Evaluation

Integrating VisualAge Generator and FlowMark
SG24-4239-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.com>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)



Printed in U.S.A.

SG24-4239-00

